First/Second Quarter, 1999

VSE/ESA™ Software Newsletter

News and information for VSE/ESA customers and vendors

G225-4508-18

Trademarks

The following are trademarks or registered trademarks of International Business Machines Corporation in the United States of America and/or other countries: AIX, AS/400, CICS, CICS Transaction Server for VSE/ESA, CICS/VSE, DB2, DFSORT, DRDA, eNetwork, IBM, Language Environment, MQSeries, Multiprise, Net.Data, Netfinity, NetRexx, OS/2, OS/390, OS/400, Parallel Enterprise Server, QMF, RAMAC, RS/6000, SQL, S/390, S/390 Integrated Server, TCP/IP for VSE/ESA, VisualAge, VM/ESA, VSE, VSE/ESA, VTAM

Domino Go Webserver, Lotus, and Lotus Domino are trademarks of the Lotus Development Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft Windows, Windows NT, and the Windows Logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

Other company, product or service names may be the trademarks or service marks of others.

About This Newsletter

The *VSE/ESA Software Newsletter* is published twice a year as a service to users of VSE systems. On the Internet, the *Newsletter* is available in PDF format (Adobe Acrobat Reader) at:
http://www.ibm.com/s390/vse/vsehtmls/newslett.htm

The information contained in this document is based on the best knowledge of the authors and on current development, has not been submitted to any formal IBM® test, and is distributed on an "As Is" basis without any warranty either express or implied. 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 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 environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead. Any performance data contained in this document was determined in a controlled environment and, therefore, the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country or not yet announced by IBM. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services.

Editor: Joel Hermann, IBM Entwicklung GmbH, Department 3257, Schönaicher Str. 220, D-71032 Böblingen, Germany (jhermann@de.ibm.com)
Subscriptions: See the subscription form at the end of the *Newsletter*.

Contents

Editor's Notes

News That's "Fit to Print"

Welcome to Issue 18 of the *VSE/ESA Newsletter*! As always, I want to express my personal thanks to all contributors. I think that this issue is a real winner, with articles from customers, vendors of VSE-related products, and IBM employees. *It's a great mix and another example of how the VSE community works together and shares information for the benefit of all.*

If you'd like to submit an article, please let me know. The rules for publishing something in the *Newsletter* are simple:

- Contact me using my e-mail address, and describe the topic that you'd like to address.
- The Editorial Board will consider the article's benefit to our readers and then either accept or reject it for the *Newsletter*.
- What should articles cover? I think the answer is: “That's up to you.” We're as open and unbiased as possible.
- Naturally, the accuracy of articles you submit is your responsibility, not IBM's. But that's business as usual.

If you are interested in contributing an article for second issue in 1999, please contact me no later than **September 1, 1999**. Thanks!

Joel Hermann (jhermann@de.ibm.com)

The IBM COBOL Family

In a recent issue of his *CICS Newsletter* (CN032299), Bob Yelavich (yelavich@ibm.net) had an item entitled, "An Excellent Reference Document for COBOL Support." Bob pointed to an appendix in an IBM White Paper which describes all of the products in the IBM COBOL Family. This document is at:

http://www.ibm.com/software/ad/cobol/wpappa.htm

It includes, for example, descriptions of these tools:

- IBM VisualAge for COBOL, Professional for OS/2 – application understanding, Year 2000 impact, program conversion
- IBM CCCA – converts source code
- IBM COBOL Structuring Facility – analysis, reporting, restructuring
- IBM CICS Application Migration Aid – converts CICS source
- IBM COBOL Report Writer Precompiler – supports Report Writer code
- IBM VisualAge for COBOL, Report Writer – supports Report Writer code
- Isogon SoftAudit/ONE and SoftAudit/2000 – inventory and assessment
- Edge Portfolio Analyzer – load library inventory
- IBM OPTI-AUDIT – inventory and assessment
- IBM VisualAge for COBOL, Test – testing
- IBM WITT Year2000 – testing
- IBM ISPF – development environment tool
- IBM TeamConnection – development environment tool
- IBM TeamConnection DataAtlas – development environment tool

As We Approach the Millennium

I had to share this one with you :-)

> "After 18 months of hard work, we are happy to inform you that we have just completed the requested project on time and on budget. We have gone through every line of code in every program in every system. We have analyzed all databases and all data files (including backups and historic archives) and modified all data to reflect the change. We are proud to report that we have now implemented these modifications to all programs and data:
>
> - Januark, Februark, March, Mak, June, Julk, August, September, October, November, December.
> - Sundak, Mondak, Tuesdak, Wednesdak, Thursdak, Fridak and Saturdak.
>
> I trust that this is satisfactork; because to be honest, none of this Y-to-K change made ank sense to us. But we understand it is a global problem, and we are glad to help in ank wak possible.
>
> Sincerelk,
> Kour IT Team
>
> PS What does the Kear 2000 have to do with it?"

Related Web Resources

VSE/ESA Home Page

Besides the *VSE/ESA Newsletter*, VSE Development also supports a VSE/ESA home page. The site is not "fancy," but it does have a wide range of information about VSE and VSE-related products from IBM and other vendors. Check out: **http://www.ibm.com/s390/vse/**

VM/ESA® Home Page

VM Development has an excellent home page with a wealth of information. This site is a *must* for VM users and is at: **http://www.ibm.com/s390/vm/**

VM and VSE Business Unit Home Page

In 1998, IBM formed a VM and VSE "business unit" in order to provide additional support to its customers with one or both of these S/390 operating systems. To learn more, access: **http://www.ibm.com/s390/vmvse/**

VSE-L Discussion Forum

VSE-L is an Internet community where VSE users gather to share tips, experiences, and support. To subscribe to this forum, which is sponsored by Lehigh University, send an e-mail to: **listserv@lehigh.edu**

In the first line of the message body, enter:
SUBSCRIBE VSE-L *your full name, your company name*

Main Technical Articles

VSE/ESA Version 2 Release 4 Available for Orders Now!

Author

Ulrich K. Ellwart
Jerry Johnston
Annette Stolvoort
VSE Technical Marketing
ellwart@de.ibm.com
gjohnston@vnet.ibm.com
ahaller@de.ibm.com

Thousands of companies all over the world rely on VSE's sturdy batch and industrial-strength online transaction processing capabilities for near 24-hours-a-day, 365-days-a-year availability. VSE/ESA Version 2 Release 4 – announced on September 29, 1998 – continues the tradition of proven, thrifty solutions by offering **CICS Transaction Server for VSE/ESA™**, new **security options**, and **IXFP/ShapShot for VSE/ESA**.

VSE/ESA Version 2 Release 4 can be ordered ***now***, and it will become generally available on June 25, 1999. If your company is located in the U.S.A. or in Europe/Middle East/Africa, you also may get VSE/ESA Version 2 Release 4 now (within the Limited Availability).

Meeting Tomorrow's Challenges

CICS Transaction Server for VSE/ESA is new with VSE/ESA V2.4. This advanced CICS® product is based on OS/390® technology.

You also have new security options with VSE/ESA V2.4. For CICS signon and transaction ID protection, a new Basic Security Manager is delivered at no extra charge. If you used CICS resource level security or if you have growing, more-demanding security needs, VSE/ESA V2.4 offers CA-Top Secret for VSE/ESA Version 1 Release 3 – a full-function security manager. CA-Top Secret is an optional, priced product shipped with V2.4. It requires CA90s Services for VSE Version 1 Release 4 (CA90s) or higher as prerequisite.

Customers who already have CA90s installed on their system need to check the required level for use with CA-Top Secret for VSE/ESA by contacting CAI, Inc. directly. Customers who do not have CA90s installed on their system must order it from CAI, Inc. by visiting:

http://www.cai.com/solutions/OS390/VSE/

Also note that tools to migrate from CICS Internal Security to either the Basic Security Manager or CA-Top Secret for VSE/ESA are available.

Doing e-business with VSE/ESA V2.4

Maybe you want to use your VSE system as a server for browser-based Internet/intranet solutions. Using VSE this way in a "two-tier" implementation leverages your investment in S/390® data, applications, skills, and hardware.

- IBM TCP/IP for VSE/ESA enables open connectivity and supports interoperability via the "sockets" programming interface. It offers common TCP/IP applications such as file transfer (FTP client and server), terminal emulation (Telnet client and TN3270 server), print distribution and serving (LPR/LPD), basic Web serving, and VSE file serving (NFS).
- CICS Transaction Server for VSE/ESA will contain the CICS Web Interface and 3270 Bridge. They allow easy Web access to your CICS applications, services, and data.[1]

You have additional options for "three-tier" solutions. For example, your Web server already may be an OS/2®, AIX®, or Windows NT system. Here your VSE system can function as an application, data, or print server. IBM has flexible middleware such as CICS Universal Clients V3, CICS Transaction Gateways V3[2], MQSeries® V2, and Net.Data® to help you implement such solutions. If you run VM/ESA, you also can exploit VM-based e-business solutions.

Other Enhancements

VSE/ESA V2.4 includes new releases of High Level Assembler, the HLASM Toolkit, and DITTO/ESA. New VSE optional products include DL/I VSE V1.11 and MQSeries V2.

VSE/ESA V2.4 supports the complete range of robust, scalable, and cost-effective S/390 servers. At the entry level is the new S/390 Integrated Server™. In the middle is the S/390 Multiprise 2000® with its reliable internal disk and modern connectivity via Open Systems Adapter 2. At the high-end is the S/390 Parallel Enterprise Server™, simply the best mainframe on the market today.

IXFP/SnapShot for VSE/ESA

If you have requirements for high performance, large-capacity, available DASD, take a look at IBM RAMAC® Virtual Array Storage (RVA). VSE/ESA V2.4 supports the unique SnapShot capability of RVA to make high speed copies of data in seconds. If you have a batch window constraint, RVA can help make your online systems available nearly around-the-clock. IXFP/SnapShot for VSE/ESA is available for VSE/ESA V2.3.1 since December 4, 1998.

DB2® Server for VSE and VM Version 6

A new version of DB2 Server for VSE and VM QMF™ became available on December 11, 1998. This new version is an optional product of VSE/ESA V2.4 and adds key new functions to the existing product set, such as DRDA® Application Requester for online applications and stored procedure support. In addition, VSE QMF and QMF for Windows now are available as optional features of the DB2 Server for VSE and VM.

---

1 The availability of the CICS Web Interface and the 3270 Bridge will be announced later.

2 CICS Universal Clients V3 and CICS Transaction Gateways V3 are part of the CICS Transaction Server.

Additional Information

These IBM Announcement Letters have more information:

- VSE/ESA V2.4 – **298-372**
- IXFP/SnapShot for VSE/ESA – **298-296**
- DB2 Server for VSE and VM V6.1 – **298-462**
- CA-Top Secret for VSE/ESA V1.3 – **298-350**

LE/VSE and DT/VSE — Performance Improvements and Support for CICS Transaction Server for VSE/ESA

Author

Peter Van Dyke
Software Designer
IBM Global Services, Perth
pvandyke@au1.ibm.com

Recent enhancements to Language Environment for VSE/ESA (LE/VSE) Release 4 can provide performance improvements for your CICS applications, especially those that run under the new CICS Transaction Server for VSE/ESA. These enhancements also provide considerable below-the-line virtual storage constraint relief (VSCR) for applications running in both the CICS and batch environments.

Debug Tool for VSE/ESA also has been modified to allow you to debug your LE-enabled COBOL, C, and PL/I applications executing under CICS Transaction Server for VSE/ESA.

These enhancements are being delivered to customers as PTFs as listed in the following table:

PTF	APAR	Component	Enhancement
UQ27971	PQ23382	LE/VSE Base	• EXEC CICS LINK performance improvements • Performance improvements for CICS AMODE 31 applications • Virtual storage constraint relief
UQ28062	PQ23385	LE/VSE COBOL	EXEC CICS LINK performance improvements for COBOL applications
UQ27875	PQ23389	LE/VSE Base	Support for DT/VSE under CICS TS VSE
UQ27876	PQ23407	LE/VSE C	Support for DT/VSE under CICS TS VSE
UQ27976	PQ23392	DT/VSE Base	Support for DT/VSE under CICS TS VSE

EXEC CICS LINK Performance Improvements

The performance of CICS high-level language programs that use many EXEC CICS LINK commands is slower under LE/VSE than in the previous language runtime environments due to the initialization and termination of a separate rununit environment (enclave/thread) for each LINKed to program.

In a CICS region with LE/VSE, each time a program issues an EXEC CICS LINK, a new LE/VSE executable environment (or rununit) is created. This involves a significant amount of resources and storage when compared with previous language runtime environments, such as VS COBOL II. VS COBOL II extends the rununit for a CICS LINK rather than create a new rununit. The LE/VSE architecture provides better program isolation and allows for the tuning of runtime options on a program-by-program basis.

The following enhancements have been made to LE/VSE in the area of initialization and termination of Language Environment enclaves and threads in order to reduce the overhead of an EXEC CICS LINK:

- Several optimizations that allow for certain resources to be reused and initialized faster.
- Faster initialization of the LE/VSE Enclave Data Block.
- Small optimizations in Stack Initialization.
- Bypass calling the global assembler user exit CEEBXITA if it is the default.
- Faster termination by allowing languages to skip Exit DSA processing.
- Several other optimizations in the initialization and termination paths.

In addition, the following enhancements have been made to LE/VSE to provide additional performance improvements for applications that run under CICS Transaction Server for VSE/ESA:

- Elimination of the CICS call to LE/VSE for Rununit Begin Invocation. LE/VSE calls Rununit Begin Invocation internally during Rununit Initialization.

  Note: The CICS trace entry for Rununit Begin Invocation is removed.
- Faster Rununit Initialization due to the merge of Rununit Work Areas (RWAs) into a single CICS GETMAIN at task initialization. CICS keeps track of the high water mark (HWM) of RWAs for a task with its first execution and uses that information on subsequent invocations to minimize RWA GETMAINs.

Performance Improvements for CICS AMODE 31 Applications

CICS regions often run with LE/VSE option ALL31(OFF) as an installation default, because the AMODE requirements of all the programs being run in the region are usually not known. This results in AMODE 31 programs being run with ALL31(OFF) unless explicitly overridden by CEEUOPT. This, in turn, results in degraded performance and excessive use of below-the-line storage for these AMODE 31 programs.

To overcome this problem, LE/VSE has been changed to detect programs that are AMODE 24 and force those to ALL31(OFF), even though the installation default is set to ALL31(ON). If CEEUOPT or assembler user exits are used to set the ALL31 option, they will continue to be used in the normal order of precedence regardless of the program's AMODE.

Note: If an installation uses dynamic calls from an AMODE 31 to an AMODE 24 program, then they must still use an installation default of ALL31(OFF) or use a specific override using CEEUOPT or Assembler user exits. AMODE 24 "autodetection" will not work for dynamically called programs.

Virtual Storage Constraint Relief

The LE/VSE phases that perform initialization and termination in the CICS, batch, and pre-initialization environments are RMODE 24. Thus they execute below the 16M line. Because of similar processing requirements, there were a large number of routines common to each of these phases. This caused these phases to be unnecessarily large and, therefore, use extra below-the-line storage.

LE/VSE initialization and termination processing has been modified, and these phases have been restructured so that most of the common routines that are capable of running above the 16MB line have been moved into the RMODE ANY phase CEEPLPKA. This results in a potential saving of approximately 738K of below-the-line SVA storage. The following table shows the changes in size of the LE/VSE initialization and termination phases:

Phase	Function	Old Size	New Size
CEECCICS	CICS Initialization/Termination	302K	46K
CEEBINIT	Batch Initialization/Termination	294K	53K
CEEPIPI	Pre-initialization	306K	65K

Debug Tool for VSE/ESA Support for CICS Transaction Server for VSE/ESA

To support fullscreen, interactive debugging of CICS applications Debug Tool for VSE/ESA needs to obtain information from CICS control blocks. With CICS Transaction Server for VSE/ESA, the layout of these control blocks has changed. The control blocks also might be in protected storage. Consequently, both DT/VSE and LE/VSE have been modified to handle the control block changes and storage protection feature of CICS Transaction Server for VSE/ESA.

Additional Information

For more information about Language Environment for VSE/ESA, go to:
http://www.ibm.com/s390/le_vse/

The IBM Debug Tool Web site is:
http://www.ibm.com/s390/dt/

For more information about MLE, see:

- IBM COBOL Family Web site
  http://www.ibm.com/software/ad/cobol/
- IBM PL/I Family Web site
  http://www.ibm.com/software/ad/pli/
- IBM VisualAge 2000 Web site
  http://www.ibm.com/software/ad/va2000/

  (contains many MLE-related links)

VSE/ESA Support for the Year 2000 – An Overview

Author

Christian Toepsch
VSE Technical Marketing
IBM Böblingen
toepsch@de.ibm.com

Today, countless newspaper articles, Web pages, books, conference proceedings, interviews and testimonies deal with the “Year 2000 Problem.” But there still seem to be a lot of people – even in Information Technology – who are not aware of the problem or believe that it can be ignored.

This is the reason why we're reprinting this article, slightly modified. It originally was part of the special Y2K issue that we did in 1996 for the *VSE/ESA Newsletter*. That issue (#13) is still available via the Web page:

http://www.ibm.com/s390/vse/vsehtmls/newslold.htm

If you haven't looked at this issue in its entirety, you should.

Enabling VSE/ESA for Year 2000

We say “enabling” VSE/ESA, because we only can fix the operating system, which is a quite small portion of all the software that runs on VSE/ESA. The simple assumption is that the problem is equally distributed over *all* software (which is pretty reasonable in this case). Thus most of what has to be done is buried in your applications and in other vendor products. **And nobody has yet found a way to isolate and fix the problem in one place, making the solution easy for all others**

But the operating system is the logical place to start, because you need a reliable test vehicle for other changes. There also is no reason to believe that fixing the operating system is much different from fixing applications. Wherever dates are handled, there is a potential hit.

Certainly, dates have different meanings, but the things that matter when fixing a date problem are:

- Is it a past- or future-oriented date?
- What are its possible values?
- How is it stored?
- Where and how is it used?

We hope that by telling you what we changed in VSE/ESA, this article may give you some hints on how to tackle other applications and make them Year 2000 ready, too.

What Are the Problems?

When you try to IPL any non-enabled VSE/ESA release with `DATE=01/01/00` – hoping it recognizes January 1, 2000 – you'll be surprised to see the system date displayed is `01/02/00`. Since century information is not shown, you can't really see where you are, but your guess is right. **It's January 1900**.

When testing non-enabled systems, we noticed a number of problems. For example, an unrecoverable loop sometimes occurred shortly after the system date became `01/06/00`. Or we saw that the date was always one day in advance until March 1st. Or when running the system without IPL, February 29th came up, even though 1900 was not a leap year. Then when we tried to IPL with `02/29/00`, the system issued an error message.

For those of you running VSE/ESA 1.3, a `19` in COMREG field SYSCENT and a `/19` behind JOBDATE give you conclusive evidence about the century. This information also is returned by the GETIME macro with the CLOCK=YES option. And some dates already can be entered with a 4-digit year or are interpreted correctly even when only 2 digits are provided. *But even with a VSE/ESA 1.3 system, many things are still missing or can go wrong.* For example:

- You cannot IPL your system with a date in 2000 and beyond, because the DATE specification only supports a 2-digit year relative to 1900.
- VSAM files (including VSAM-managed SAM files) cannot have expiration dates in 2000 and beyond, because the expiration year is stored with 2 digits only and is always taken relative to 1900.
- Other files can have explicit expiration dates in 2000 and beyond, but the checks for expiration ignore the century; and the date calculation for a retention period across the century transition always yields an early date in 1900.

  Also, files with an expiration date of 99/365 actually expire and may be silently deleted on December 31, 1999, although that's normally not what you want.
- Recovering or restoring a DL/I data base may fail, because the year in backup and journaling timestamps has only 2 digits and is always taken relative to 1900.

These are just a few of the known problems. Other problems may occur that we don't even know about, because incorrect date handling – just like a programming bug – may fail in ways that nobody anticipates.

What Are the Fixes?

This section only discusses some basic rules that VSE/ESA Development agreed on and followed in our implementation. This methodology was necessary, since fixing Year 2000 problems usually involves several people; and you don't want to end up with multiple solutions.

Let's start with a basic observation: **The fact that the last two digits uniquely identify the year of a date, if the range of possible dates is within 100 years, doesn't suddenly change across the century transition.** What changes is only the fact that the century cannot any more be implied to be a constant `19`. Instead, it can be determined by a 100 year window that crosses into 2000 and beyond.

For the VSE/ESA operating system, we made extensive use of this "sliding window" technique, because all the dates we handle today are safely within a 100 year

range and will continue to be so for at least another 30 years from now, if we let the range move along with the current date. A sliding window allowed us to maintain compatibility for date formats with a 2-digit year that are used in several places within the operating system, just by fixing the *interpretation* of these dates. And that was already our first basic rule: ***Maintain compatibility.***

An interface to convert a 2-digit year to 4 digits with a sliding window is provided by Language Environment® for VSE/ESA and can be used in your COBOL for VSE and PL/1 for VSE programs, as well as from High Level Assembler. We did not use this interface for the operating system itself, because it wasn't available in all possible circumstances where we needed it. We developed, therefore, our own service, accessible via a new Assembler macro **YEAR224**. You may use it as well.

Using the sliding window technique you really have 99 years to play with, and you can pick the 100 year range that fits best a certain date. You can specify how far to go into the past and how far to go into the future. In the VSE/ESA operating system, most dates are past-related; and we used the range `(-79,+20)` for those. But there are exceptions. A file expiration date, for example, is future-oriented and must be handled differently. In this case we used a `(0,+99)` range relative to the creation date. This window actually slides with each creation date, rather than with the current date.

User friendliness was our second basic rule. Input dates always can be entered as a 2-digit year, not only for compatibility but also for convenience. We (VSE) can figure out what is meant. But optionally, you also should be able to enter it with 4 digits, if clarity is what you prefer.

For output dates in messages and formatted output, we have opted for 4-digit dates only. This is certainly user friendly, although it sounds like bad news for tools that process dates in such output. We thought about it and concluded that tools ought to like that as well, because they get complete information right away, rather than something that needs to be interpreted again.

Enabled VSE/ESA Releases

All currently supported releases of VSE/ESA are Year 2000-enabled. Current releases are:

- VSE/ESA 1.4.4
- VSE/ESA 2.3.2
- VSE/ESA 2.4.0

VSE/ESA 2.4.0, which was announced with a limited availability, will be generally available in June.

Year 2000 support was provided via PTFs and integrated into the refreshes for VSE/ESA 1.4.3 and VSE/ESA 2.2.0. *We strongly recommend that you install the latest refresh level, not the single PTFs.*

Testing for the Year 2000

The intent of an early, Year 2000-enabled VSE/ESA system is, of course, to allow you to develop and test your own changes with the appropriate system interfaces in place and without having to worry about system problems. IPLing the system with a date in 2000 and beyond is easy, and doing it just to “see what happens” is a tempting idea. There are, however, some risks involved in such an experiment:

- All your existing data with expiration dates before 99/365 is considered expired and may be overwritten without warning. This risk is unique to the enabled system. On a non-enabled system, century information is not considered. Therefore, all such data is considered unexpired.
- All your logs – in particular those that safeguard the integrity of your databases – may get polluted by entries out of time sequence, making any subsequent recovery difficult or impossible. The interpretation of such data for auditing purposes is also in question. Similar problems may be expected with any accounting data you collect.
- If your system is connected to a network, date inconsistencies may be propagated to other systems and extend the scope of potential damage.

In addition to such foreseeable effects, you must be prepared for unpredictable results produced by applications that have not yet been adapted. Even a more controlled form of exploitation requires careful consideration of the overall environment. For example, with an enabled VSE/ESA system it is now safe to use expiration dates beyond 99/365, since century information is taken into account. However, if you are sharing data with a non-enabled system, such data is again exposed to being overwritten without warning, since that system will consider all such data as expired.

This is not meant to discourage the use of an enabled system for its intended purpose, but rather to point out that careful testing for Year 2000 requires isolation and dedicated resources. This is most easily achieved under VM, where subsetting of resources is most flexible. You need to have the TODENABLE function active to allow an IPL with a different date for a certain user. LPAR exploitation is the solution of choice for a native system. If none of these solutions is available, an additional inexpensive S/390 configuration, like an S/390 Integrated Server, might be a viable alternative.

Additional Information

The VSE and Year 2000 home page has information that is specific to VSE. It's located at:

http://www.ibm.com/s390/vse/vsehtmls/vse2000.htm

The IBM Year 2000 home page is at:

http://www.ibm.com/year2000/

Please note that IBM offers only Year 2000-ready hardware and software products and will offer service for most of these products until at least January 31, 2001. In addition, IBM has developed an extensive portfolio of offerings, tools and resources to help you execute your own Year 2000 readiness plans.

The IBM Year 2000 Web site provides a vast array of information to assist you, and IBM will continue to expand Internet-based customer support. The Year 2000 site is updated frequently so you can directly obtain information such as:

- How to prepare your PCs for the Year 2000.
- Extensive information about the best ways to test your systems.
- Readiness status of all IBM products.

For the balance of 1999 and in the weeks following January 1, 2000, many customers will request assistance on Year 2000 issues. As customer-support demands increase, so will our capabilities to respond – but the magnitude of this potential increase is unknown. In anticipation of this growing demand, IBM is developing contingency plans to address customer needs, add capacity at call centers and reassign employees as necessary. IBM intends to provide the best possible service and support to customers with warranties and service agreements. The first priority must be to satisfy existing agreements, so IBM customers who do not have contractual agreements for maintenance and support may encounter delays for requested assistance.

IBM emphasizes that the best way to minimize potential problems is to make Year 2000 readiness a top priority, if you haven't already done so. IBM urges you to migrate to ready products and solutions if your business processes currently rely on not-ready products. IBM also encourages you to install the latest maintenance level for ready products to take advantage of functional and readiness updates.

The IBM Year 2000 Web site is the primary source of information about what IBM offers customers in the way of information, products, services and support. Please visit this site frequently as you prepare for Year 2000.

DOS/VS COBOL and VS COBOL II Programs and the Year 2000

Authors

Alice Crema
Tom Ross
IBM COBOL Family
IBM Santa Teresa Laboratory
crema@us.ibm.com
tmross@us.ibm.com

There are no known problems with older IBM COBOL products related to the Year 2000 that will affect your applications. There are really only two issues to be concerned with for COBOL applications and the Year 2000:

1. Date-related logic
2. Service support from IBM

Note: The `DATE-COMPILED` and `WHEN-COMPILED` features of DOS/VS COBOL will return incorrect 4-digit year dates, but these are probably not used in business logic.

You can get service support for DOS/VS COBOL and VS COBOL II programs indefinitely by using the Language Environment run-time library with just a run-time migration. This is described in "Service Support for Programs Compiled with DOS/VS COBOL or VS COBOL II" on page 15.

DOS/VS COBOL is going out of service on December 31, 1999; and VS COBOL II (VSE feature) went out of service in April 1998. As a result, an application with `NO DATE-RELATED LOGIC` could run in a supported environment beyond the turn of the century by only a run-time migration, with *no* source code changes necessary.

Repeat: *No source code changes, and your DOS/VS COBOL and VS COBOL II programs can be running with full IBM service support and no Year 2000 problems if they are running under Language Environment and have no date-related logic problems!*

Date-Related Logic

Of course, an application that DOES have date-related logic must be examined to see if it has a Year 2000 problem. If it does, then programs will have to be changed, which means at the very least a compile, link and test.

At this point, you can combine source upgrade (compiler migration to COBOL for VSE/ESA) with Year 2000 conversion. IBM does not provide any way for DOS/VS COBOL programs to obtain a 4-digit year date, so you must compile with at least COBOL for VSE/ESA in order to use IBM-provided services for obtaining a 4-digit year date.

Service Support for Programs Compiled with DOS/VS COBOL or VS COBOL II

IBM will continue to provide service support for the execution of programs compiled with the DOS/VS COBOL and VS COBOL II compilers, as long as these programs are using the Language Environment for VSE/ESA (5686-094) version of the COBOL library routines.

Every COBOL program requires library routines in order to execute. They may be statically linked to the phases or dynamically accessed at execution time. If the library routines that are being used are supported by IBM service, then you can call IBM when there is a problem and get the problem fixed.

For example, the library routines for DOS/VS COBOL programs exist in the DOS/VS COBOL run-time library, the VS COBOL II run-time library, and in the Language Environment run-time library. If your DOS/VS COBOL programs are running using the DOS/VS COBOL library, then after December 31, 1999, they are not supported by IBM service. If your DOS/VS COBOL programs are running using the Language Environment run-time library, then your programs are supported by IBM service and will be supported well into the 21st century.

For DOS/VS COBOL phases or VS COBOL II phases compiled with `NORES`, you need to link edit with Language Environment to use the new library routines. For VS COBOL II phases compiled with `RES`, then you can just make the Language Environment library routines available at execution time in place of the VS COBOL II library routines by using LIBDEF.

Source Conversion NOT Required to Complete Migration

The ideal state is that programs should be compiled with a supported compiler (recommended is COBOL for VSE/ESA) running with a supported run-time library (Language Environment for VSE/ESA). However, this ideal state can be reached gradually in a fully supported manner:

1. Run-time migration, followed OPTIONALLY at some future date by
2. Compiler migration.

What If I Need to Change a DOS/VS Program?

The DOS/VS COBOL compiler will not be supported after December 31, 1999, but the programs that were generated by DOS/VS COBOL will be supported under Language Environment.

Once you have completed the run-time migration, if a change needs to be made to a COBOL program for maintenance or enhancement, you only need to run the source through a source conversion tool such as the COBOL and CICS/VS Command Level Conversion Aid (CCCA, 5686-A07) or the Conversion feature of VisualAge COBOL Enterprise and then compile using the COBOL for VSE/ESA compiler. This way you can gradually convert your source with a minimum cost to your company.

Additional Information

For complete information on both run-time migration and source conversion, see the publication, *COBOL/VSE Migration Guide*, GC26-8070. Presentations also available for viewing on the Web under "library" at:

http://www.ibm.com/software/year2000/

In addition, there are the IBM redbooks:

IBM COBOL and Language Environment for VSE/ESA – How to Upgrade Now, SG24-4277, and

Taking Advantage of Language Environment for VSE/ESA, SG24-4798.

IBM redbooks can be viewed and ordered on the Web at:

http://www.ibm.com/redbooks/

VSE/ESA 2.4 Security – Miscellaneous Topics

Author

Klaus-Dieter Wacker
VSE Development
IBM Böblingen
kdwacker@de.ibm.com

VSE/ESA 2.4 introduces a new security concept. Resource Managers (like CICS Transaction Server for VSE/ESA) delegate all security checks to a central **Security Manager**. This article discusses miscellaneous topics for implementing security with VSE/ESA 2.4.

Editor's Note

The last issue of the *VSE/ESA Newsletter* (#17) had two articles dealing with security for VSE/ESA 2.4. They are available on the Web at: **http://www.ibm.com/s390/vse/vsehtmls/vsenew17.htm**. Look for the files **vsebsec.pdf** and **vsebsts.pdf**. This article completes the overview to this topic.

More on Basic Security Manager (BSM)

Password Handling

A password is the only way the system can verify if a user really is the person she/he claims to be. Therefore, the password belongs solely to the user and must be kept secret under all circumstances. The BSM supports that view.

When an administrator defines a user, a temporary password is assigned. It permits initial entry to the system, at which time the person required to choose a new password. Unless the user divulges it, no one else knows the user ID/password combination. If a user forgets the password, the administrator has to assign a new temporary password, which must be changed at the next signon.

Revoke Date

An administrator can prevent a VSE user from entering the system by assigning a "revoke date" in that person's User Profile. A revoke date is useful when a user is to be prevented from entering the system from a certain date. For example, let's say that you have students who require access to VSE during the semester until August 31. It is then practical to set a revoke date to September 1st.

System Revoke

The system also can revoke a user. When a hacker tries a known user ID with different password, the BSM monitors the invalid signon attempts. When a certain limit is reached, the current date is set in the user's revoke date field, which prevents signon of the user at all.

The default limit of invalid signon attempts is five. You can tailor this to your needs in the IESELOGO skeleton. During VSE/ESA startup, the program IESIRCVT is executed (procedure USERBG) which sets the revoke limit in the BSM control table. For more information on this, see the *VSE/ESA Administration Guide* for VSE/ESA 2.4

Note: A user ID which is revoked needs manual intervention from the administrator before being used again. With the User Profile Maintenance (UPM) Dialog, the administrator can change the revoke date in the user's profile, either by clearing the date or by specifying a future date.

What About an External Security Manager?

The BSM, provided by IBM, is part of VSE/ESA 2.4 and supplies basic security functions. If you have more comprehensive security requirements, you can use an External Security Manager (ESM) which has the functionality you want. VSE/ESA 2.4 provides the System Access Facility (SAF) for seamless integration of a RACROUTE-based ESM.

CA-Top Secret for VSE/ESA

CA-Top Secret for VSE/ESA is a software product licensed from Computer Associates International.

CA-Top Secret 1.3 provides enterprise security, management and control of mainframe and networked environments with integrated security solutions that protect your information assets. It provides flexibility and control through a unique hierarchical security configuration similar to the organizational structure of your corporate environment. Administrative tools, reporting options, and automatic logging capabilities accompany CA-Top Secret, securing your environment while enabling comprehensive auditing and controlled sharing of data resources.

CA-Top Secret for VSE/ESA 1.3 requires CA90 Services for VSE (1.4 or higher) as a prerequisite. CA-Top Secret is shipped with VSE/ESA 2.4. It is key protected and requires a unique user access key. That key is generated and delivered by IBM when CA-Top Secret is explicitly ordered.

If started without valid key, CA-Top Secret runs in "demo mode." When a security violation occurs, it will be reported with a console message only.

Single User Definition

To ease the administration effort for user registration, a single user definition is desirable. Both BSM and ESM support that requirement.

Segment Concept

The Security Manager (SM) keeps more than user settings which are required for signon checking. In addition, it offers a "segment concept," where various settings for resource managers could be kept. In the CICS segment, for example, all user settings relevant to CICS TS for VSE/ESA are stored. (See Figure 1 on page 19.) When a user logs on to CICS TS, this information is requested from the SM. This makes use of a Signon Table (SNT) obsolete in CICS TS for VSE/ESA.

The RACROUTE interface provides a standard EXTRACT call to retrieve this information.

```
IESADMUPCI               ADD OR CHANGE CICS SEGMENT
Base     II       CICS     ResClass ICCF

   OPERATOR ID..............  SYA   Enter 3 character id for user HUGO
   OPERATOR PRIORITY.......  000   Operator priority between 0-255
   XRF SIGNOFF.............  2     Sign off after XRF takeover (1=yes,2=no)
   TIMEOUT.................  30    Minutes until sign off between 0-60

   PRIMARY LANGUAGE........        National language for CICS messages


     Place an 'X' next to the operator classes for this user

   01 X    02 _     03 _     04 _     05 _     06 _     07 _     08 _
   09 _    10 _     11 _     12 _     13 _     14 _     15 _     16 _
   17 _    18 _     19 _     20 _     21 _     22 _     23 _     24 _



PF1=HELP                    3=END                    5=UPDATE
PF7=BACKWARD   8=FORWARD
```

Figure 1. UPM Dialog – User Profile Editor/BSM (CICS Segment)

The same is true for the Interactive Interface (IES segment). When you use an ESM, the Interactive Interface signon program extracts the user's data from the ESM. When the ESM supports the Interactive Interface segment (CA-Top Secret does) and you have defined an IES segment in the ESM, no User Profile record has to exist for this user in the VSE/ESA Control file. If the IES segment extract fails, however, a user profile must exist in the VSE/ESA Control file from where the Interactive Interface signon program gets the user's settings.

Signon to CICS TS Application Driver

Most of a VSE customer's applications are driven by CICS TS for VSE/ESA. This section discusses methods of performing signon to a CICS TS application driver. All signon methods inherit features offered by the Security Manager, such as:

- Password checked for expiration,
- New password checked against password rules, and
- Revoke processing.

VSE Interactive Interface

The VSE Interactive Interface is the preferred way to sign on to a CICS TS for VSE/ESA application system. After successful signon, the "Menu Selection" of the VSE Interactive Interface is presented. To present the VSE Signon screen after terminal power-on, specify `IEGM` as the CICS TS good morning transaction (DFHSIT: GMTRAN=IEGM).

Further characteristics apply:

- Password change by user
- Single signon (one user can signon once)
- Logon here possibility

Native CICS TS

You also can use the native CICS TS signon program (transaction CESN) to control user signon (Figure 2). After successful signon, the user gets a CICS command line. Even if you prefer this method, the VSE administrator should use the Interactive Interface to maintain the user profiles.

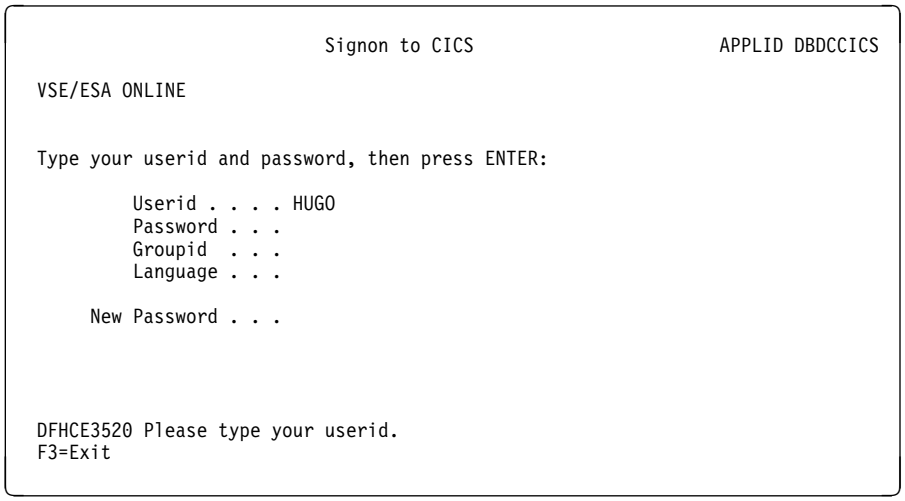


Figure 2. CICS TS Signon Screen (CESN)

Further characteristics apply:

- Password change by user
- Language selection for CICS TS messages
- Multiple signons possible

Private Signon

When the VSE-provided signon methods are not suitable for some reason, you can develop your own signon program. The technical basis for this are the security-related CICS TS API commands:

SIGNON This command associates the security capabilities and operator characteristics of the specified user with a terminal. It allows to specify a new password. The feedback information of the Security Manager is passed back via error conditions.

SIGNOFF Signoff a user from a terminal.

In this case, the VSE administrator also should be able to use Interactive Interface. Other characteristics can apply:

- Password change by user (strongly recommended to implement)
- Single/multiple signon (depends from the application requirements)

VSE/ESA 2.4.0 and CICS Coexistence

VSE/ESA 2.4 allows you to run the CICS Transaction Server (CICS TS) together with CICS/VSE® 2.3. Thus these two versions of CICS can "coexist" on the same VSE/ESA 2.4 system.

Common User List for CICS TS and CICS/VSE

The standard setup for CICS/VSE 2.3 in a VSE/ESA 2.4 system requires a DFHSNT table to let CICS/VSE check the user authority during signon (CESN transaction). The administrator has to create an SNT entry for each user.[3]

The disadvantage with this approach is that users defined in SNT cannot change their password during signon. This may lead to the situation that passwords differ on the CICS Transaction Server and the CICS/VSE system. However, there is an acceptable solution for that problem as outlined in the following paragraphs.

The CICS/VSE setup can be changed so that the signon check is delegated to the Security Manager (either BSM or ESM) in VSE/ESA 2.4. The Security Manager checks the user's authorization against the security data stored, allows password changes, and even applies "user revoking" to users signing on to CICS/VSE. The CICS user characteristics (like transaction security keys or timeout) need to be specified in a default SNT entry which applies for all users not explicitly listed in the SNT table. (The security exit programs support signon authorization only. They do not support resource checking.)

Installation: The following steps are required:

1. The phases DFHXSE and DFHXSSCO implement the interface to the Security Manager. These phases are distributed in the CICS TS product library (default: PRD1.BASE). Make the phases accessible for CICS/VSE 2.3:

   a. Rename the original CICS/VSE DFHXSE phase in library PRD2.CICSOLDP.

   b. Be sure that PRD1.BASE is contained in the LIBDEF search chain of CICS/VSE 2.3.

2. Define DFHXSSCO to CICS/VSE via the command:

   ```
   CEDA DEF PROG(DFHXSSCO) GROUP(VSESPO) LANG(ASS) RES(YES) RSL(PUBLIC)
   ```

   Shared CSD file

   When the CSD file is shared between CICS TS and CICS/VSE, you must issue the CEDA command from the CICS Transaction Server.

3. Define an default SNT entry as shown below. The default entry must follow all other entries. Use skeleton DFHSNTCO in ICCF library 59 for defining the entry.

   ```
   * DEFAULT ENTRY FOR ESM CONTROLLED USERS (LAST ENTRY IN SNT)
            DFHSNT TYPE=(ENTRY,DEFAULT),                    *
                  OPIDENT=OPR,                              *
                  OPPRTY=4,                                 *
                  SCTYKEY=(4,5,10,24,37,60,61,62,63,64)
   ```

4. Activate the External Security Manager for CICS/VSE via entry `EXTSEC=YES` in DFHSIT. Use skeleton DFHSITCO in ICCF library 59 for defining the entry.

5. Restart CICS/VSE

   When initialization of the CICS/ESM interface during CICS/VSE startup is complete, the following message is issued:

[3] This does not apply for customers using an External Security Manager with CICS/VSE 2.3.

```
DFH3532I EXTERNAL SECURITY INTERFACE ACTIVE
```

Interaction: Signon to CICS/VSE is performed with the CICS-supplied transaction CESN (Figure 3).

```
CICS Signon


DFH3560I TYPE IN YOUR USERID AND OPTIONALLY PASSWORD

          Userid   ==>
          Password ==>
          Language ==>
(*)   New Password ==>
```

Figure 3. CICS/VSE Signon Screen (CESN)

Note: The New Password line shows up only when entry EXTSEC=YES is coded in DFHSIT. For users who have an explicit SNT entry, input in the New Password line has no effect. (Sign-on is performed without further notice.)

Migration Considerations

Security-Related Resources to Be Migrated

1. Interactive Interface user profiles from an old VSE Control File.
2. ICCF user records in DTSFILE.
3. CICS user profiles from a CICS/VSE signon table (SNT).
4. Transaction definitions. In previous VSE releases, these where contained in the CICS/VSE table PCT.
5. For VSE batch security users, DTSECTAB.

Perform Migration Task

Use the VSE migration utility IESBLDUP to migrate user profiles from the VSE Control File, the CICS SNT (applicable for BSM only), and the ICCF DTSFILE. For more information, see *VSE/ESA System Utilities* for VSE/ESA 2.4

Warning

Never simply restore an older VSE Control File on VSE/ESA 2.4, since the record layout is not compatible.

Use the CICS Security Migration Aid (SMA) to migrate other security-related resource definitions. SMA has a two-stage process. For more information, see the *CICS/VSE Security Migration Guide*.

Stage 1 Collects security definitions, which currently reside in numerous CICS tables, into a central repository (VSAM dataset DFHXSMA).

Stage 2 Generates the definitions for Security Manager out of the resource records in DFHXSMA.

Users of VSE batch security should drop private user definitions from DTSECTAB and define all users solely in the VSE Control File (except for the system-provided definitions for users DUMMY and FORSEC).

Additional Information

For further information about security, refer to the manual *VSE/ESA Administration* for VSE/ESA 2.4.0.

Global Variables for VSE/ESA Job Control

Author

Frank Munzert
VSE Development
IBM Böblingen
munzert@de.ibm.com

In 1995, a WAVV User Group requirement asked VSE Development to provide *global SETPARM variables*. For the sake of compatibility with job control terminology, I will use the term *symbolic parameter* instead of *SETPARM variable* in this article. The adjective *global* relates to the *scope* of symbolic parameters and is discussed in the following sections.

Scope of Symbolic Parameters

In VSE/ESA releases up to VSE/ESA 2.3, the scope of symbolic parameters is restricted to a job. The set of symbolic parameters is released during end-of-job processing and has to be rebuilt (if applicable) by the next job. In this context *released* means that GETVIS storage related to symbolic parameters (organized in subpools named **IJBPRCnn**, where **nn** identifies the partition) is given back to the system via FREEVIS.

In VSE/ESA 2.4, two additional sets of symbolic parameters are introduced. The first contains parameters which are valid for the duration of a whole VSE/POWER job (or for the lifetime of a dynamic partition). The second contains parameters valid for the whole system, which last until the system is IPLed again.

Thus we have three different parameter pools with different lifetimes:

1. (DOS) job parameters, which last until end-of-job (DOS EOJ) or until end-of-procedure, if defined in a cataloged procedure.

   They are called *symbolic parameters at level n*, where *n* is the nesting level. n=0 if the parameter is defined in the job stream and 1≤n≤15 if the parameter is defined in a cataloged procedure.

   There are independent parameter subpools for each of the 16 nesting levels and for each partition. Passing parameters (via EXEC PROC) means copying parameter entries from one subpool to another.

2. VSE/POWER job parameters, which last until * $$ EOJ processing (POWER EOJ).

   They are called *symbolic parameters at POWER job level*. Independent POWER job parameter subpools exist for each partition.

3. System parameters, which last unit the system is shut down or re-IPLed.

   They are called *symbolic parameters at system level*.

Symbolic parameters at VSE/POWER job or system level are defined exclusively by means of the SETPARM command. There is no difference if you define them in a job or in a cataloged procedure.

Symbolic parameters at level n can be defined with the SETPARM, PROC or EXEC PROC statement. They can be passed from one nesting level to another.

Search Sequence

If a symbolic parameter is used in the operand field of a job control command or statement, its value is retrieved according to the search sequence shown in Figure 4.

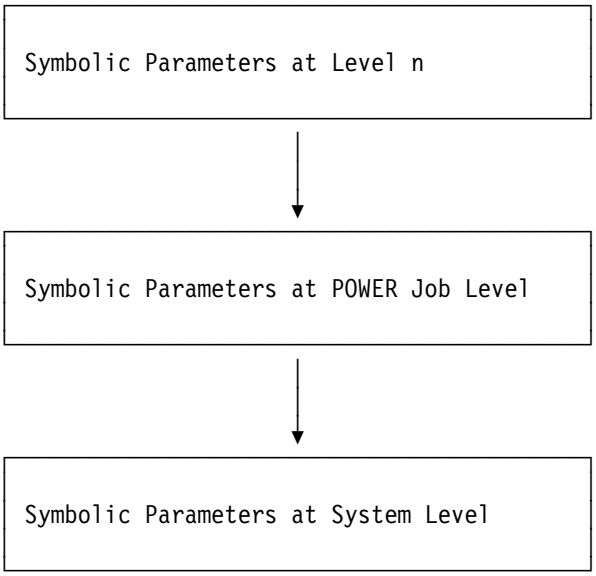


Figure 4. Search Sequence for Substitution of Symbolic Parameters

If the symbolic parameter is not found in any of the three parameter pools, message `1F3nD` is issued.

For example, if the statement `// ASSGN SYS005,&CUU` is encountered on nesting level 4, then the system searches the following three pools for a substitution of `&CUU`:

1. symbolic parameters at level 4,
2. symbolic parameter at POWER job level
3. symbolic parameter at system level

If a user program (or JCL exit routine) resolves a symbolic parameter by means of macro GETSYMB, the same search sequence is in effect. A return code of X'10' in register 15 indicates that the symbolic parameter was not found in any of the three parameter pools.

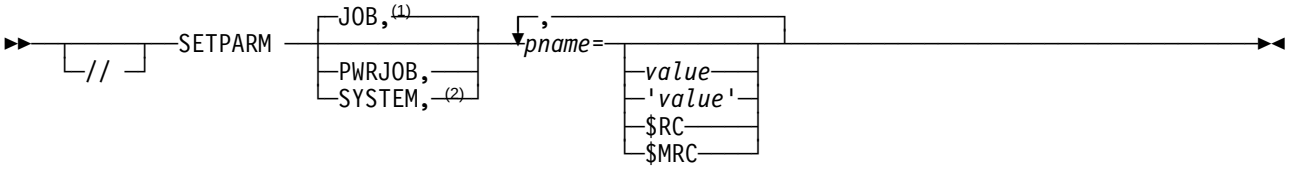
Note: The GETSYMB macro is enhanced to support

- AMODE 24 or 31
- RMODE 24 or ANY.

If a parameter name is specified in the condition expression of an IF statement, the same search sequence is in effect when retrieving this parameter's value.

Command Syntax

A new operand for the SETPARM command can be used to specify which kind of symbolic parameter is to be defined (Figure 5).



Notes:

1 SETPARM JOB is only accepted within a job.

2 SETPARM SYSTEM is only accepted if issued from the BG partition.

Figure 5. SETPARM – JCC, JCS Format

JOB

Means that you want to specify symbolic parameters at level n, which are valid for the (DOS) job or cataloged procedure currently active. They are cleared at end-of-job time (processing of /&) or end-of-procedure time (processing of /+), respectively. Only symbolic parameters at level n can be manipulated with the EXEC PROC or PROC statement.

PWRJOB

Means that you want to specify symbolic parameters at POWER job level, which are valid for the VSE/POWER job currently active. They are cleared at POWER EOJ time (processing of * $$ EOJ). There is no difference if you specify SETPARM PWRJOB in a job or in a cataloged procedure.

SYSTEM

Means that you want to specify symbolic parameters at system level, which are valid for all partitions during the lifetime of the system. They are cleared when the system is shutdown or (re-)IPLed. There is no difference if you specify SETPARM SYSTEM in a job or in a cataloged procedure.

Sample Scenario

The following sample scenario illustrates how symbolic parameter values are retrieved according to their search sequence. For the scenario, I've made the following assumptions:

1. BG's JCL startup procedure ($0JCL.PROC) contains one SETPARM statement:

   ```
   // SETPARM SYSTEM,CUU1=120          1
   ```

2. The profile procedure of dynamic class Y (STDPROF.PROC) contains one SETPARM statement:

   ```
   // SETPARM PWRJOB,CUU1=130,CUU2=160    2
   ```

3. There is a cataloged procedure called ABC.PROC containing the following statements:

   ```
   // ASSGN SYS005,&CUU1
   /+
   ```

The following VSE/POWER job is processed in dynamic partition Y1:

```
* $$ JOB JNM=POWERJOB,CLASS=Y,DISP=D
// JOB DOSJOB1
// SETPARM CUU1=170,CUU3=181          3
// EXEC PROC=ABC
// ASSGN SYS005,&CUU1
// ASSGN SYS006,&CUU2
// ASSGN SYS007,&CUU3
/&
// JOB DOSJOB2
// ASSGN SYS005,&CUU1
// ASSGN SYS007,&CUU3
/&
* $$ EOJ
```

This job's output on SYSLST may look as follows:

```
// JOB DOSJOB1
// SETPARM CUU1=170,CUU3=181
// EXEC PROC=ABC
// ASSGN SYS005,130                    Substitution 1
EOP ABC
// ASSGN SYS005,170                    Substitution 2
// ASSGN SYS006,160                    Substitution 3
// ASSGN SYS007,181                    Substitution 4
EOJ DOSJOB1
// JOB DOSJOB2
// ASSGN SYS005,130                    Substitution 5
// ASSGN SYS007,&CUU3                  Substitution 6
1F34D  PARAMETER CUU3    NOT DEFINED
EOJ DOSJOB2
```

Explanation:

Substitution 1 It is nesting level 1, and &CUU1 is not defined as symbolic parameter at level 1. Next, the system searches the pool of symbolic parameters at POWER job level. Because of **2**, &CUU1 becomes 130. If cataloged procedure ABC had been invoked via `// EXEC PROC=ABC,CUU1`, then &CUU1 would have been 170 (because of **3**).

Substitution 2 It is nesting level 0; and in **3**, &CUU1 was defined as symbolic parameter at level 0. Thus &CUU1 becomes 170.

Substitution 3 It is nesting level 0, and &CUU2 is not defined as symbolic parameter at level 0. Next, the system searches the pool of symbolic parameters at POWER job level. Because of **2**, &CUU2 becomes 160.

Note that if POWERJOB had been submitted to a static partition, &CUU2 would not have been found in any of the three pools, resulting in message 1F34D.

Substitution 4 It is nesting level 0; and in **3**, &CUU3 was defined as symbolic parameter at level 0. Thus &CUU1 becomes 181.

Substitution 5 It is nesting level 0, and &CUU1 is not defined as symbolic parameter at level 0. Next, the system searches the pool of symbolic parameters at POWER job level. Because of **2**, &CUU1 becomes 130.

Note that if POWERJOB had been submitted to a static partition, &CUU1 would have been substituted by 120 (because of 1).

Substitution 6 It is nesting level 0, and &CUU3 is not defined as symbolic parameter at level 0. Next, the system searches the pool of symbolic parameters at POWER job level, not found. Finally, the system searches the pool of symbolic parameters at system level, not found. There is no substitution, therefore, and error message 1F34D is issued.

Storage and Capacity Considerations

For both new parameter pools, up to 10 2KB-chunks of GETVIS storage are allocated to store symbolic parameter information records. This GETVIS storage is allocated with the attributes LOC=ANY and SPACE. For a dynamic partition, the required storage areas are taken from the dynamic space GETVIS area. For static partitions (especially for symbolic parameters at system level), they are allocated in the system GETVIS area, preferably 31-bit. The GETVIS pool identifier is IJBPRC, concatenated with a partition identifier. You can use the GETVIS command to display information related to system GETVIS storage allocated for symbolic parameters (Figure 6).

```
==> GETVIS SVA,DETAIL
    AR 0015 SUBPOOL       REQUEST  <--SVA-24-AREA---    --SVA-ANY-AREA-->
    AR 0015 IJBPRC0010    SPACE                  8K                  16K
    AR 0015                         00242000-00242FFF    02C06000-02C09FFF
    AR 0015                         00276000-00276FFF
```

Figure 6. Sample Display of GETVIS Pools Related to Symbolic Parameters (BG only)

Note that the GETVIS chunks below the 16MB line contain both control blocks for procedure handling and information records for symbolic parameters at level n. The ones above the 16MB line contain information records for symbolic parameters at POWER job level and symbolic parameters at system level.

It takes n+10 bytes to store a symbolic parameter information record, where n denotes the length of the character string assigned to the parameter name. For n=50, you may store up to 329 symbolic parameters in each new pool; for n=1, up to 1846 of them.

Consider the following scenario:

```
// SETPARM SYSTEM,PARM1='OLD VALUE'        1
...
// SETPARM SYSTEM,PARM1='NEW VALUE'        2
...
// SETPARM SYSTEM,PARM1='OTHER VALUE'      3
```

Observe that the lengths of the character strings in 1 and 2 are both equal to 9, whereas in 3 this length is 11. Let's assume that PARM1 has not been defined so far. Thus 1 creates a 19-byte symbolic parameter information record for PARM1.

During processing of 2 , the system recognizes that PARM1 is already defined. Since the lengths of *OLD VALUE* and *NEW VALUE* are equal, the system does not create a second symbolic parameter information record for PARM1. It just replaces the value information in the already existing symbolic parameter information record.

During processing of **3** , the system creates a second 21-byte symbolic parameter information record for PARM1 and nullifies the first one. The storage occupied by nullified symbolic parameter information records is not available to store other symbolic parameter information records.

Additional Information

For further information about global variables for VSE/ESA job control, refer to the manual *VSE/ESA Administration* for VSE/ESA 2.4.0.

Implementing TCP/IP Printing on VSE

Author

John Lawson
*Intell*Ware Systems, Inc.
lawson@intelliware.com

The TCP/IP for VSE/ESA product offers the VSE user the capability to implement several TCP/IP application functions. Two of these, **Line Printer Requester** (**LPR**) and **Line Printer Daemon** (**LPD**), allow VSE systems to participate as a client and server with other heterogeneous systems in network printing.

As an LPR client, VSE can issue requests to network print servers to print VSE data and batch-generated reports. As an LPD server, VSE can act as a network print server and process print requests from LPR clients in the network. This article describes the implementation of both of these facilities and some of the pitfalls that can be encountered in doing so.

VSE as an LPR Client

The LPR client support in TCP/IP for VSE/ESA gives the VSE user more capabilities to distribute printed data directly to network users. VSE printed output can be directed to network printers located in user departments on local area networks (LANs) or to printers in remote locations connected by wide area networks (WANs). Small remote locations with only one or two users could even use the Internet or a dial connection to access their VSE system and retrieve reports. One of our clients did exactly this with a single PC, an attached printer and an Internet connection. Figure 7 is a pictorial representation of these three environments.

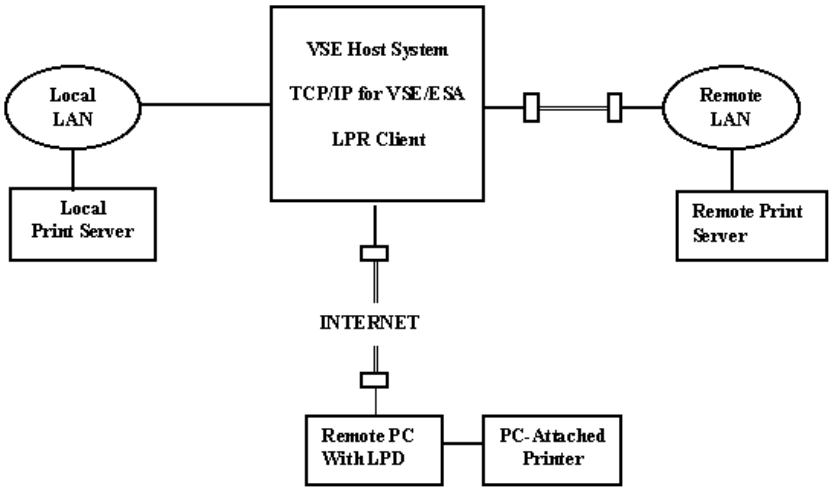


Figure 7. Sample VSE LPR Client Network

The VSE LPR client can be invoked from a batch jobstream or interactively from a CICS terminal session. VSE/POWER LST queue entries also can be automatically printed by the LPR client using a facility of TCP/IP for VSE/ESA called *automatic LPR*.

A Line Printer Daemon (LPD) must be running on the print server to which the VSE LPR client issues print requests. Several network servers have the LPD function supplied as part of their operating system. However, these may need to be configured if TCP/IP and LPR have not been used for printing.

Other network hosts that are used as print servers may require the purchase of server software to support LPD. LPD software can be found on some of the shareware Web sites such as:

http://www.shareware.com or
http://www.tucows.com

Some network printers such as the HP printers with JetDirect attachments have the LPD function built into the hardware and only require configuration from a PC.

TCP/IP for VSE/ESA must be configured and active in a partition for the VSE LPR clients to function. Network connection and routing definitions need to be correctly defined to support access to the print servers to which the VSE LPR clients will issue print requests. Entries specific to LPR and LPD support are described below.

The VSE CICS LPR Client

The LPR client can be invoked interactively from a CICS terminal by entering a CICS transaction called *LPR*. CICS transaction and program definitions are implemented as part of the TCP/IP for VSE/ESA installation to support LPR and other TCP/IP applications from a CICS terminal. Figure 8 on page 32 and Figure 9 on page 33 show the terminal input and responses from a sample CICS LPR client session. Commands entered from the terminal are in bold for clarity.

```
LPR
TCP200I Client -- Startup --
TCP207I Copyright (c) 1995-1997 Connectivity Systems Incorporated
TCP202I Attempting to Establish Connection
TCP204I Connection has been Established
Client manager connection Established.
LPR Ready:
set host=hpprint1
LPR Ready:
set printer=local
LPR Ready:
cd prd2.config
Change has completed.
LPR Ready:
set cc=no
LPR Ready:
print ipinit00.l
Opening the file for transfer
Establishing connection with LP Daemon
Request the queue for output
Checking LP daemon availability
Transferring the data file
More...
```

Figure 8. CICS LPR Client – Screen 1

The CICS LPR client session is invoked by entering the LPR transaction. Several commands supplied with TCP/IP for VSE/ESA are used to control the LPR session.

The SET command has several options. The SET HOST command identifies the name or IP address of the remote print server. If a name is used, it must be defined in the VSE TCP/IP configuration. The following configuration command is used to define a name for a print server and its IP address:

```
DEFINE NAME,NAME=HPPRINT1,IPADDR=200.200.200.10
```

The SET PRINTER command identifies the name of the print queue on the print server. Some print servers don't use print queue definitions. Thus any print queue name would be valid. Some print servers use lowercase names for print queues and require the CICS terminal to be set to lowercase to allow entry of a valid print queue name.

The CD command causes TCP/IP for VSE/ESA to set the "VSE directory path" to PRD2.CONFIG. The PRINT command causes the specified member to be sent to the print server for printing.

The SET CC command controls whether standard carriage control characters should be handled or ignored. Figure 9 on page 33 shows the responses received when the request is completed.

```
Data file has transferred correctly.
Transferring the control file
Control file has transferred correctly.
Closing the connection with LP Daemon
Connection complete
LPR Ready:
quit
TCP201I Client -- Shutdown --
TCP205I Connection Complete -- Already Closed
```

Figure 9. CICS LPR Client – Screen 2

The QUIT command is used to terminate the LPR session and the LPR transaction so that the terminal can be used for other CICS transactions.

Other commands that can be used in an LPR session include:

- PWD – Print current Working Directory
- DIR – List current directory
- LONG – Long format listing of queued print files
- SHORT – Short format listing of queued print files
- QUERY OPTIONS – Display current settings
- EXEC name – Execute a script of commands from library member **name.L**

There are also several additional options for the SET command besides the more common ones used in the example in Figure 8 on page 32. These include:

- SET INSERTS – name of a phase with printer control characters
- SET JOBNAME – jobname on LPD's separator pages
- SET TITLE – title on LPD's title page
- SET DISP – POWER DISP after printing (KEEP|DELETE|HOLD)
- SET COPIES – number of copies to print
- SET FCB – name of FCB phase for forms control
- SET NOEJECT – eliminate initial form feed
- SET MAIL – email address for LPD notification if LPD supports it
- SET TRANSLATE – name of alternate translate table
- SET CRLF – =OFF eliminates CR/LF delimiter; =UNIX replaces CR/LF with LF
- SET DELREQ – controls deletion of print file if LPD supports it
- SET USER – used by TCP/IP security routines
- SET PASSWORD – used for LPR security
- SET RECFM, SET LRECL, SET BLKSIZE – specifications for printing disk files

LPR Scripts

LPR commands that are frequently used can be combined into a script and executed using the LPR EXEC command or the automatic LPR function. The script of commands is catalogued in a VSE library as source member **name.L**. For example, to execute a script catalogued as LPRSCRIPT.L, enter the command EXEC LPRSCRIPT during the LPR session. The name of the script member must be defined in the TCP/IP configuration with the following command:

```
DEFINE NAME,NAME=LPRSCRIPT,SCRIPT=LPRSCRIPT
```

Figure 10 on page 34 shows a script file that would produce the same results using just the EXEC command as those entered in the sample LPR session in Figure 8 on page 32 and Figure 9.

```
// JOB  LPRCONFG  CATALOG LPR SCRIPT FILE
// EXEC LIBR
  ACC S=PRD2.CONFIG
  CATALOG LPRSCRIPT.L       REPLACE=YES
SET HOST = HPPRINT1
SET PRINTER = LOCAL
CD PRD2.CONFIG
SET CC = NO
PRINT IPINIT00.L
QUIT
/+
/*
/&
```

Figure 10. LPR Script Example

Controlling LPR Printers

Network printers are different than VSE mainframe printers in that special control character sequences are used to control print formatting (page orientation, font size, and lines per inch, for example). TCP/IP for VSE/ESA supports sending these control characters with an assembled phase called the *INSERTS* phase. An INSERTS macro is used to define what hexadecimal control characters are to be sent with the printout sent by an LPR request. The LPR client identifies the name of the INSERTS phase to use with the SET INSERTS command.

Figure 11 shows an example jobstream to assemble an INSERTS phase.

```
// JOB  LPRCTRL  ASSEMBLE LPR PRINTER CONTROL PHASE
// LIBDEF *,SEARCH=PRD1.BASE
// LIBDEF PHASE,CATALOG=PRD2.CONFIG
// OPTION CATAL
// EXEC ASSEMBLY
LPRCTRL  INSERTS  DEFINE,                                              X
               HEADER=1B451B266C314F1B2831551B2873307031362E363768382E3X
                      576307330623054,                                 X
                      TRAILER=1B45
         END
/*
// EXEC LNKEDT
/*
/&
```

Figure 11. Sample INSERTS Phase Assembly

The characters in the HEADER parameter are sent at the beginning of the report, and those in the TRAILER parameter are sent at the end of the report. Control characters can also be sent at the start of each page using a PAGE parameter. The assembly in this example will generate a phase named *LPRCTRL*.

The VSE Batch LPR Client

The TCP/IP batch LPR client is initiated using a batch jobstream. The same commands used in the CICS LPR client can be coded in the jobstream used for the batch LPR client. The responses from the batch LPR client are written to SYSLST.

Figure 12 is an example of a batch jobstream that would perform the same LPR request as the CICS LPR example in Figure 8 on page 32 and Figure 9 on page 33.

```
// JOB  LPRJOB   EXAMPLE TCPIP LPR JOB
// LIBDEF *,SEARCH=PRD1.BASE
// EXEC CLIENT,PARM='ID=00,APPL=LPR'
SET HOST=HPPRINT1
SET PRINTER=LOCAL
SET CC=NO
CD PRD2.CONFIG
PRINT IPINIT00.L
QUIT
/*
/&
```

Figure 12. Batch LPR Client Example

The parameters on the VSE execute statement specify that an LPR client function is to be invoked and identify the ID of the TCP/IP partition that is to process the LPR requests.

Automatic LPR

An LPR client can be invoked automatically by entries written to the VSE/POWER LST queue. TCP/IP for VSE/ESA can be configured to monitor a specific class in the LST queue and initiate the LPR client when the disposition of LST queue entries in that class are set to D or K. The following DEFINE command in the TCP/IP configuration will cause POWER LST queue class `X` to be monitored for LPR eligible output:

```
DEFINE EVENT,ID=LST_LISTEN,TYPE=POWER,CLASS=X,QUEUE=LST,ACTION=LPR
```

The USER, DEST and UCS parameters on the POWER LST card provide the information that the TCP/IP partition needs to perform the LPR operation. Figure 13 on page 36 is an example of a batch jobstream that will cause automatic LPR to be invoked. Two different ways of coding the LST card are shown.

```
* $$ JOB JNM=LPRJOB,CLASS=0,DISP=D
*
* $$ LST CLASS=X,DISP=K,DEST=(*,LOCAL),USER=HPPRINT1,             X
               UCS=HPCTRL
                                  - or -
* $$ LST CLASS=X,DISP=K,DEST=(*,LOCAL),USER=LPRSCRIPT
*
// JOB  LPRJOB   EXAMPLE TCPIP LPR EXAMPLE
// EXEC LIBR
 ACC S=PRD2.CONFIG
 L IPINIT00.L
/*
/&
* $$ EOJ
```

Figure 13. Automatic LPR Example

The USER parameter identifies either the name or IP address of the print server. The USER parameter also can be the name of a script file containing any of the LPR commands described earlier.

The DEST parameter identifies the name of the print queue, if any, on the print server. Print queue names on some servers may be more than eight characters or defined in lowercase, in which case a script file must be used.

The UCS parameter can be used to define the name of an INSERTS phase, if one is required. The SET INSERTS command in a script file also can be used instead of this parameter.

Figure 14 lists the console messages produced when an automatic LPR session is invoked. The last message will indicate failed rather than completed if any part of the LPR process has an error. The automatic LPR support will retry the print request a second time. If it fails again, the POWER LST queue entry is set to a special disposition type of **Y**. The SET DIAGNOSE=LPR command will produce more detailed console messages that are useful in debugging a problem where the LPR request has failed.

```
F8 0083 0026: TCP911I Processing Event:LST_LIST Type:POWER Action:LPR
F8 0083 0026: TCP907I Processing Output:LPRJOB  ,09568 for automated
F8 0083 printing at:LPRSCRIPT         on:LOCAL
F8 0083 0026: TCP908I Processing Output:LPRJOB  ,09568 has completed
```

Figure 14. Automatic LPR Console Messages

VSE as an LPD Server

TCP/IP for VSE/ESA also can be configured for VSE to act as a network print server. Network users can issue LPR requests to VSE to print their files on VSE-controlled printers or to store their print files in VSE libraries or files defined to the TCP/IP file system. An LPR client is required on the network user's system to issue requests to the VSE LPD server. Some platforms come with an LPR client that operates from a command prompt in linemode. There are also graphical LPR clients available through the Web sites mentioned on page 31.

The options on linemode LPR clients vary by platform. A typical LPR command may look like the following:

```
lpr -p printer -s server printfile.name
```

The `server` parameter identifies the IP address or name of the VSE TCP/IP system running the LPD server. The `printer` parameter identifies the name of a print queue defined in the TCP/IP for VSE/ESA partition's startup configuration. The `printfile.name` parameter is the name of the file the user wants the VSE LPD to print.

Line printer daemons and print queue definitions are defined in the TCP/IP for VSE/ESA startup configuration with the DEFINE LPD command.

Some modified examples of the DEFINE commands that are supplied with TCP/IP for VSE/ESA and sample linemode LPR client commands that would be used to print to them follow.

1. This entry defines a print queue on VSE called FAST.

   ```
   DEFINE LPD,PRINTER=FAST,QUEUE='POWER.LST.A'
   ```

   LPR print requests to this print queue are written to POWER LST queue A. A "public" entry with directory name of POWER is required in the TCP/IP partition's file system configuration. This type of LPD definition requires sufficient GETVIS storage in the TCP/IP for VSE/ESA partition to receive the entire print file before it is written to the POWER LST queue. It should only be used for processing small print files.

   The LPR client would issue the following command to print a file named `config.sys` on this VSE LPD 'printer':

   ```
   lpr -p fast -s vsehost config.sys
   ```

   `vsehost` in these examples is a name defined for the VSE host system in a network domain name server (DNS). A DNS associates names with IP addresses.

2. The second example defines a VSE LPD print queue called FASTLIB.

   ```
   DEFINE LPD,PRINTER=FASTLIB,QUEUE='PRD2.PRINT'
   ```

   LPR print requests to this print queue are written to a VSE sublibrary PRD2.PRINT. A "public" entry with directory name PRD2.PRINT is required in the TCP/IP file system definitions. This type of LPD entry would be used to store print files in VSE libraries.

   The LPR client command to print the config.sys file on this VSE LPD "printer" would be:

   ```
   lpr -p fastlib -s vsehost config.sys
   ```

3. The last example defines a VSE LPD print queue called LOCAL.

   ```
   DEFINE LPD,PRINTER=LOCAL,QUEUE='POWER.LST.A',LIB=PRD2,SUBLIB=TEMP
   ```

   This would be the preferred definition to use for receiving LPR requests for large print files. The received print file is staged in a VSE sublibrary PRD2.TEMP before writing it to POWER LST queue A. A "public" entry is still required for POWER. However, one is not required for the staging library. The entry in the VSE library is deleted after it is transferred to the POWER LST queue.

The LPR client command to print the config.sys file on this VSE LPD "printer" would be:

```
lpr -p local -s vsehost config.sys
```

Print files are stored on VSE using the jobname transmitted by the LPR client. If none is provided, then the name is generated by TCP/IP for VSE/ESA in the form `LPDFAnnn`, where `nnn` is the job number transmitted by the LPR client. Print files sent by LPR clients typically do not have the standard printer carriage control characters that VSE generated print files have. LPR print files will be printed as normal text files without carriage control.

Additional Information

In this article, we have looked at the facilities TCP/IP for VSE/ESA provides to enable VSE to be an LPR client and a network LPD print server. By understanding how these facilities are implemented, VSE users can continue to expand their VSE system's participation in the client/server world.

The IBM manual *TCP/IP for VSE/ESA User's Guide* (SC33-6601) provides additional details on the implementation of LPR and LPD facilities. PTFs available for the TCP/IP for VSE/ESA product also include documentation updates which should be reviewed for new functions and enhancements to the LPR and LPD functions.

*Intelli*Ware focuses on products and services exclusively for the worldwide VSE community and can be reached through the following:

Telephone: **1-800-4-VSEESA** or **817-277-0800**
Electronic access:
via email: **info@intelliware.com**
via WWW: **http://www.intelliware.com/**

The VSE/ESA Navigator Function Prototype

Authors

Ingo Franzki
Jörg Schmidbauer
VSE Development
IBM Böblingen
ifranzki@de.ibm.com
jschmidb@de.ibm.com

This article is one example of how Java and TCP/IP can be used with VSE/ESA. It describes a prototype that we call the **VSE/ESA Navigator Function**, which is the result of three diploma theses at the *Berufsakademie Stuttgart*, finished last September. It was implemented by Ingo Franzki (now VSE development), Ivonne Kessler (now working for a German company) and Martin Smolny (now Merva development).

The client/server application consists of:

- A Java-based client and
- A server that is implemented in C for VSE/ESA running in a VSE partition.

Client and server communicate over a TCP/IP socket connection. It is possible to be connected to multiple VSE hosts at a time, and one server can handle multiple concurrent clients.

The benefits are obvious. Since the client side is a Java application, it can run on any Java-enabled platform, like Windows, OS/2, AIX, or OS/390. TCP/IP makes the communication setup very easy.

The client side provides:

1. A Java-based graphical user interface (GUI) for VSE/ESA, and
2. A Java-based programming interface (API) to extend the GUI's capabilities.

The server side:

1. Handles all requests from multiple clients, and
2. Provides a C-based programming interface (API) to extend the server's capabilities and to provide further service functions for new client functions.

The overall structure is illustrated in Figure 15 on page 40.

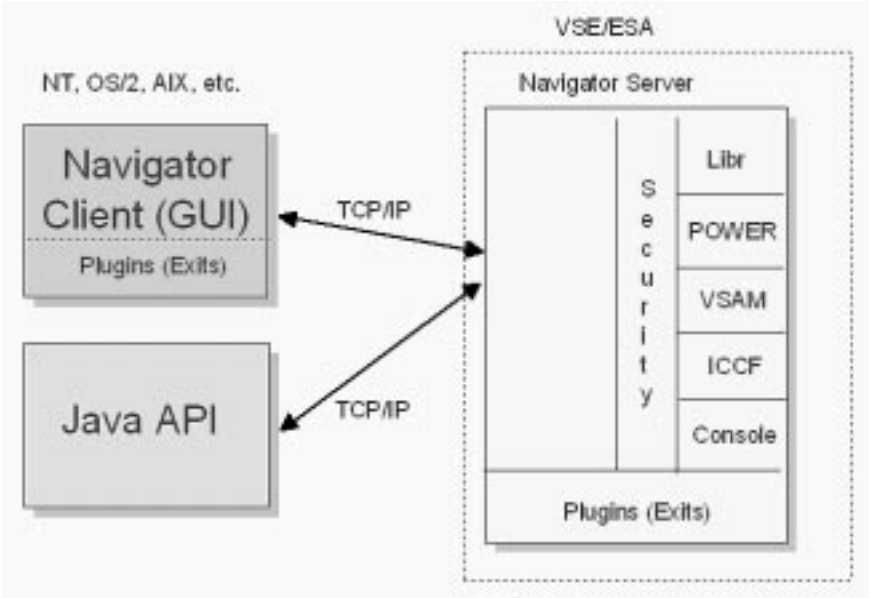


Figure 15. Overview

The Navigator server runs in a VSE batch partition, is written in C for VSE/ESA, and handles TCP/IP communication with multiple Navigator clients. Security is provided by a logon mechanism to the Navigator server. Figure 16 shows the application's main window.

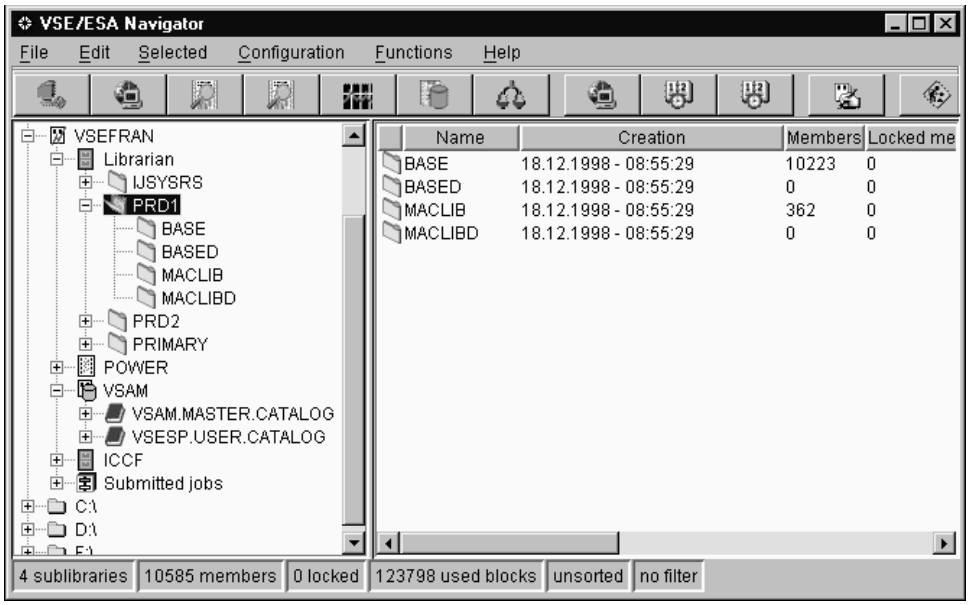


Figure 16. VSE/ESA Navigator Main Window

Supported VSE File Systems

The Navigator can access the following VSE file systems.

Librarian Read/write. Copy members, create/delete members, download/upload members, full text search, graphical file compare.

VSAM Read only. Mapping of VSAM records to (spreadsheet-like) tables, filtering of records, display of VSAM catalogs and clusters, export of VSAM data in spreadsheet format.

ICCF Read only. Download of members, copy into a VSE library, submit as job to VSE/POWER, full text search.

POWER Read/write. Display job output, submit/release, delete, copy to/from Librarian, full text search, graphical file compare.

Local Read/write. Access to any local or LAN-based hard disk.

Access to the VSE file systems is enhanced through a:

- **Search function** that allows one to locate specific files within the various file systems and to search for specific text strings within files.

  For performance reasons, the search is done on the host. Only the search results are sent to the client.

- **Graphical file compare function** that allows use of any platform-dependent compare tool (like WINDIFF.EXE for Windows platforms). Any file – host-based or local – can be compared to any other file.

  Host files are downloaded to the client before they are compared.

- Support for displaying VSAM file data in readable form (Figure 17). So-called **map files** can be created for any VSAM cluster (except VRDS) through a dialog for mapping the record structure to a relational view. The data can be exported to a spreadsheet file.

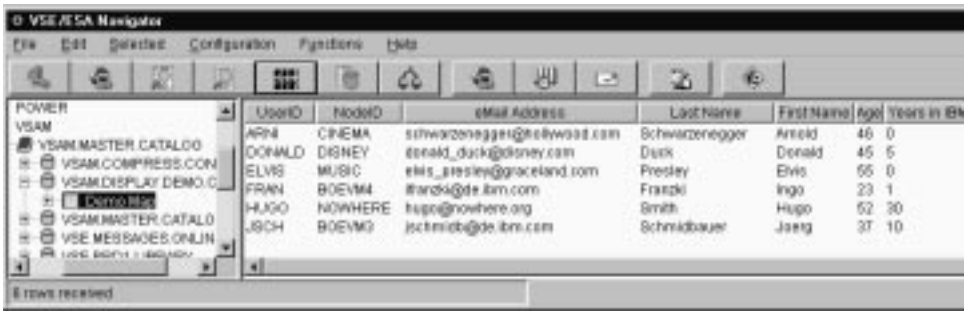


Figure 17. Displaying VSAM Files

In many cases, not all fields of a data record are important for a certain user or user group. Thus for each map, multiple *views* can be defined to describe a *subset* of the fields contained in the map.

All maps and views are stored in one single VSE library member in readable form, so that they can be accessed by all Navigator users.

Concepts

Plugins

Client plugins Most client dialogs of the VSE/ESA Navigator are written as plugins. This means that new dialogs can be added to the application when required.

The resulting Java class files are simply copied into the plugin directory within the Navigator's directory tree. They are detected automatically during the next Navigator startup. An example for a simple plugin is part of the Navigator package.

Server plugins Host plugins are implemented in C for VSE/ESA. Each plugin consists of a VSE phase, which must be defined in a server configuration member that resides in a VSE library. All defined server plugins are loaded during the next server startup. For example, the *Display System Activity* function (“Display System Activity” on page 46) consists of one client and one server plugin.

File Handling

One major implementation for file handling allows for the association of file types with applications. It's the same principle as you know from PC file systems.

If an application is associated with a certain file or VSE member type, these files or members will show a related menu choice in their pop-up menu (Figure 18 on page 43). Double-clicking on a file or member invokes the primary application of this file. For example, if an HTML document is stored in a VSE library, a simple double-click downloads it and loads it into the preferred Web browser. For program sources, different editors and tools can be associated to operate on these files.

For all associated applications, it can be specified whether the changed file contents are to be uploaded to the host after work on the document is finished.

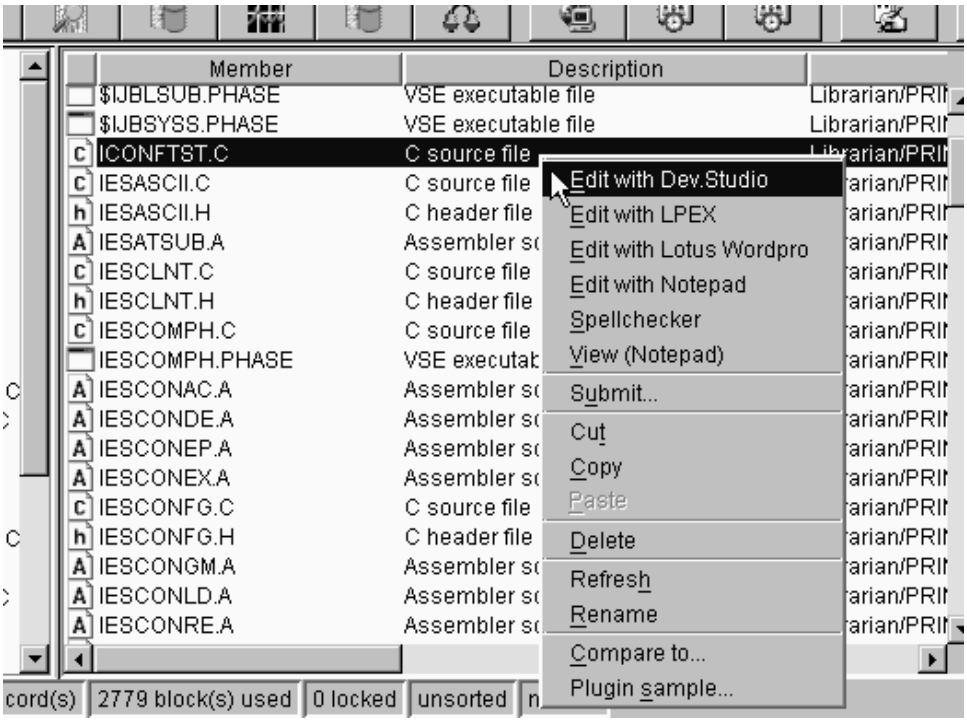


Figure 18. Associating Applications with File or Member Types

Drag and Drop Support

Drag and drop support is provided for all supported file systems in both directions – from client to server and vice versa and from one VSE host to another one (“Copy Files From One VSE Host to Another” on page 49).

Note that it is not possible to drag files from inside the Navigator main window to any outside application. This is a Java 1.1 restriction. However, all local and LAN-based drives can be accessed from within the Navigator main window. In effect, therefore, drag and drop can be performed from any source to any target, except for dragging files into open applications (like spreadsheet or text processing tools).

Security

Security is implemented on the server side only. In case of VSE/ESA 2.4, user ID and password are checked using the currently active VSE security manager. Also, access rights to VSE resources are verified.

For VSE/ESA 2.3, only logon security is provided. User ID and password are checked directly against the VSE.CONTROL.FILE. Additionally, the server supports maintenance of a list of user IDs and IP addresses that are allowed to connect to the server.

API Functional Overview

The Java API consists of a set of Java class files that provide methods for accessing VSE file systems. For example, the Librarian API supports:

- Getting a list of VSE libraries from connected host,
- Getting a list of sublibraries of a given library,
- Getting a list of members with a given sublibrary,

- Copying a member from one sublibrary into another one,
- Renaming, deleting, downloading a member,
- Uploading a file into a VSE sublibrary,
- Creating or deleting a sublibrary,
- Getting the properties of a library,
- Searching for members and strings within members,
- Copying an VSE/ICCF member to a VSE library,
- Copying a VSE/POWER queue entry into a VSE library, and
- Copying a library member to a VSE/POWER queue.

There are similar APIs for POWER, ICCF, VSAM, and for the operator console. These API classes can be used independently of the Navigator GUI.

GUI Functional Overview

This section describes some selected VSE-specific functions. The main goals here are to provide a point-and-click interface to VSE functions and to combine two or more separate functions to get additional value.

System Console

This dialog (Figure 19) offers the functionality of a user console, like the Interactive Interface console dialog.

To make it easier to repeatedly enter similar commands, the console dialog offers a so-called *autotext* function. Using autotext, a format string (like *msg fb,data=%*) can be specified. Thus, for example, each security server command entered could prefixed with the same prefix string. The percent sign is replaced by the command entered in the upper text field.

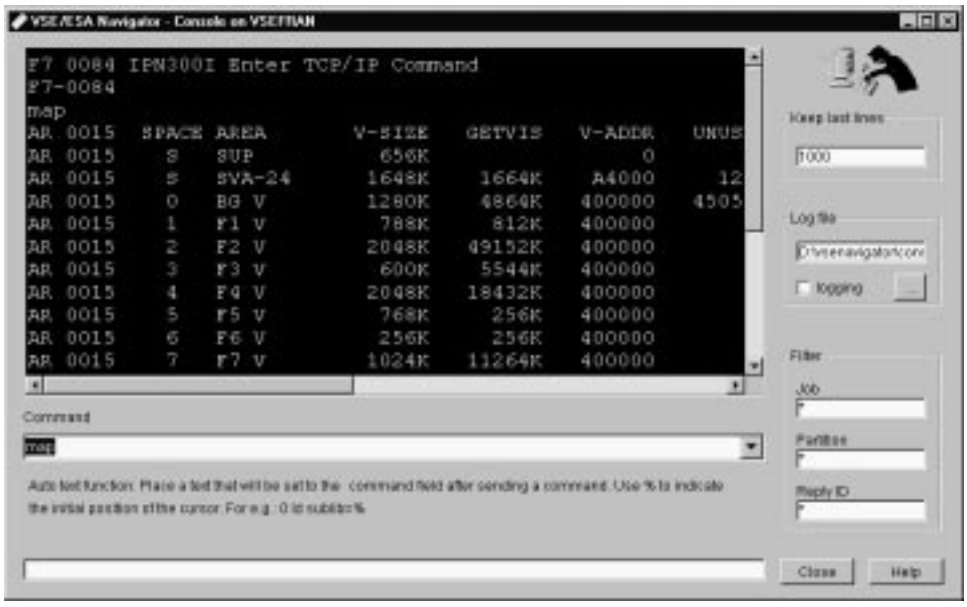


Figure 19. Console Function

Compile Helper Plugin

This function can be used to scan compile output in the VSE/POWER LST queue for keywords that indicate compiler messages. Of course, these keywords are freely configurable. Such lines, together with a couple of lines before and after a compiler message, are then automatically downloaded to the client. From hundreds of pages output, only those lines that are really of interest to the user are shown. For example, Figure 20 is an extract of a compiler output with more than 10 segments with 100 pages each.

```
...
Found keyword in COMPILE.16667[1] at Line 9:
// EXEC EDCCOMP,SIZE=EDCCOMP,PARM='LONGNAME RENT SS SOURCE SHOWING...

Found keyword in COMPILE.16667[4] at Line 2853:
 1100       |#if !defined(WIN32)
 1101       |    struct timezone   tz;
===========> ....a.............................................
*=ERROR=====> a - EDC0044 AN INCOMPLETE STRUCT OR UNION TAG C...
 1102       |#endif  /* !WIN32 */
 1103       |#ifdef WIN32
 1104       |    time_t    timetime;
...
Found keyword in COMPILE.16667[5] at Line 401:
EOJ COMPILE   MAX.RETURN CODE=0012
...
```

Figure 20. Compile Helper Output Text

Retrace Products and PTFs

This dialog (Figure 21 on page 46) can be used to retrieve a list of all installed products and PTFs on a specific VSE host. It submits two MSHP jobs to retrace all products and PTFS that are installed on the host.

Output of the jobs is combined into one history file display. The data is then kept locally on the client to speed up access to this data later.

The data can be refreshed at any time by clicking on the *Refresh* button.

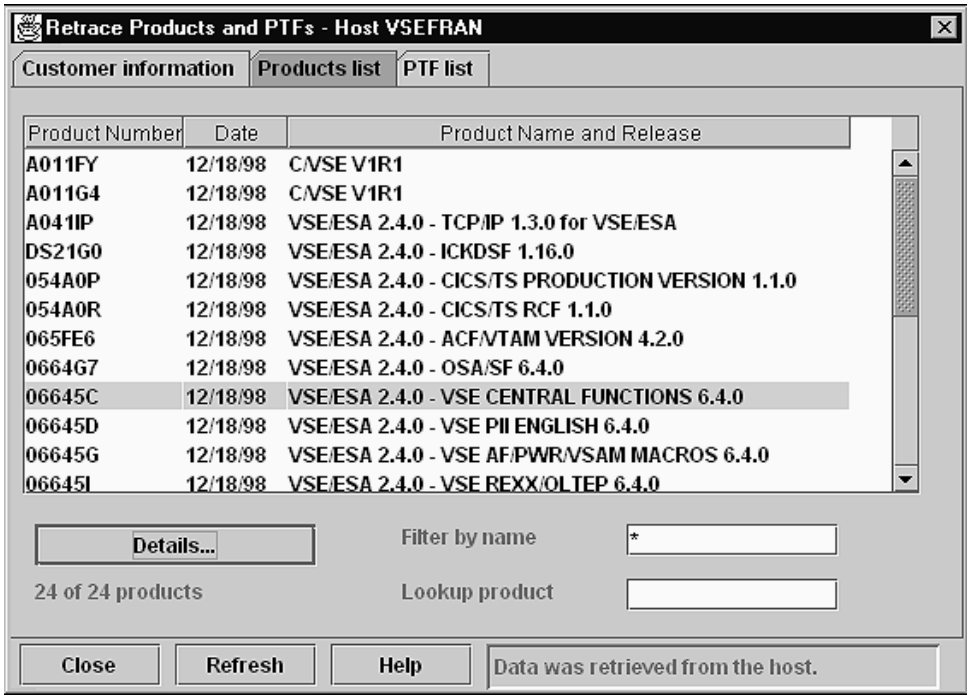


Figure 21. Retrace Products and PTFs

Products and PTFs can be easily sorted just by clicking on a column header within the products or PTFs table. A PTF can be looked up by starting to type the PTF number in the "Lookup PTF" text field. The table scrolls to the PTF automatically. Details can be displayed for all products and PTFs.

Display System Activity

This function (Figure 22 on page 47) collects various data about the current system load. The display is separated into:

- General data (like overall CPU load, paging data, and CICS-related data),
- Activity in static partitions,
- Activity in dynamic partitions/classes,
- Channel and device activity,
- Common Turbo Dispatcher-related activity,
- CPU-specific data (like total CPU time per CPU, spin time, and non-parallel time).

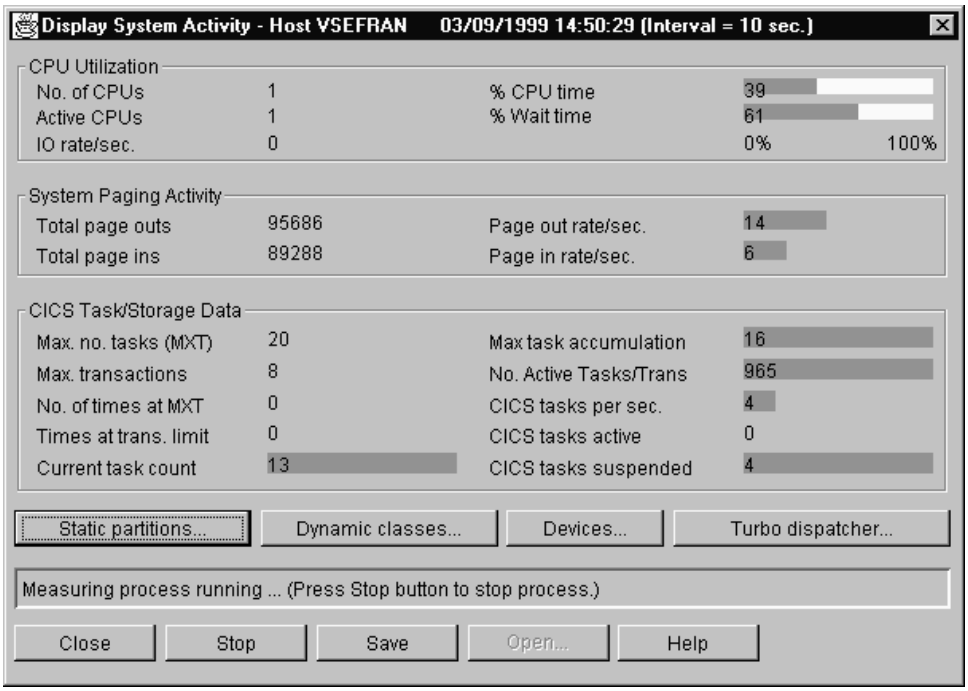


Figure 22. Display System Activity Function

The data is collected by starting the *IEXM* transaction, which supports measurement of system activity in time intervals of at least 10 seconds. The IEXM data is preformatted by the Navigator server and sent to the requesting client in the specified measurement intervals.

Turbo Dispatcher-related activity data is retrieved through a dispatcher exit routine. It can be monitored through the dialog box shown in Figure 23 on page 48.

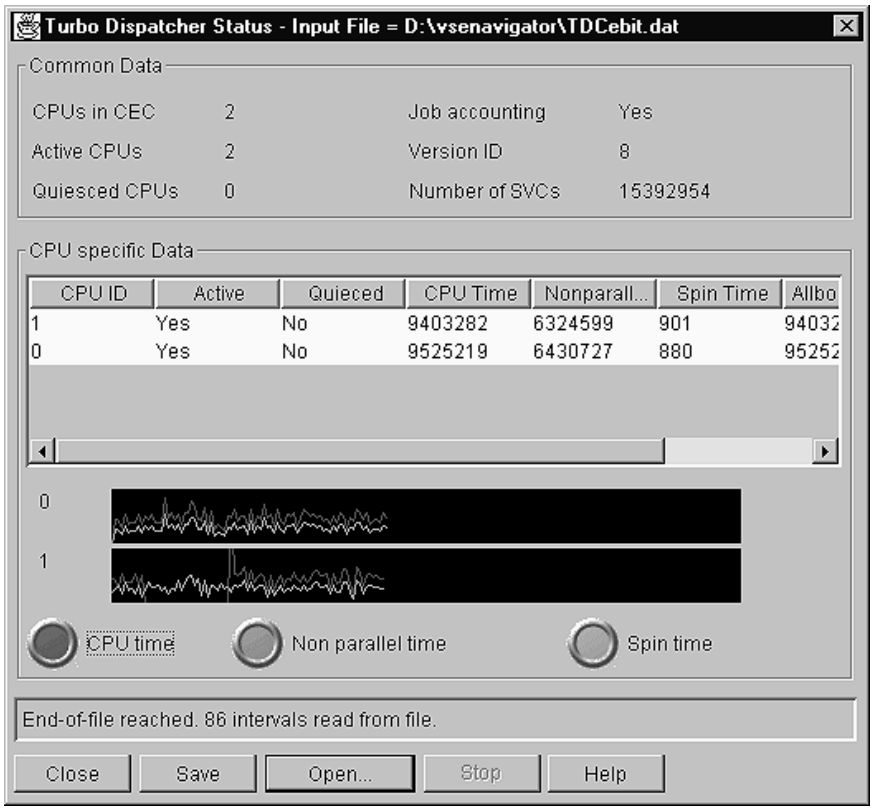


Figure 23. Turbo Dispatcher-Related Data

Common data appears in the upper half of the box. For each active CPU, one chart is drawn to show the CPU-specific activity curve.

Jobs running in static partitions are monitored through the display in Figure 24.

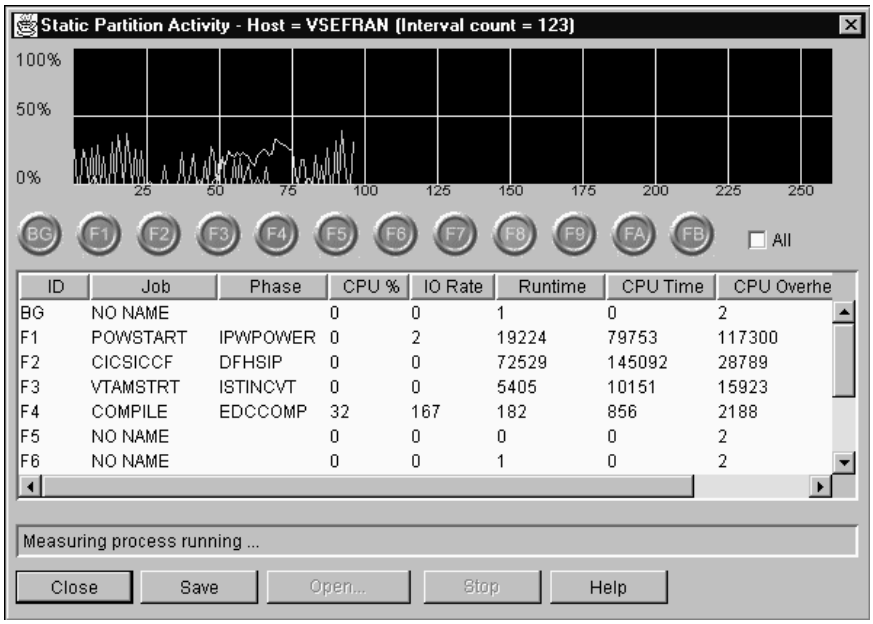


Figure 24. Static Partition Activity

All data is automatically collected up to a limit that is specified by the user. It can be saved to disk in simple spreadsheet format.

These output files can be read in again later to replay the activity data with faster or lower speed. They also can be read by spreadsheet applications like Lotus 1-2-3 to create charts or formatted reports (Figure 25).

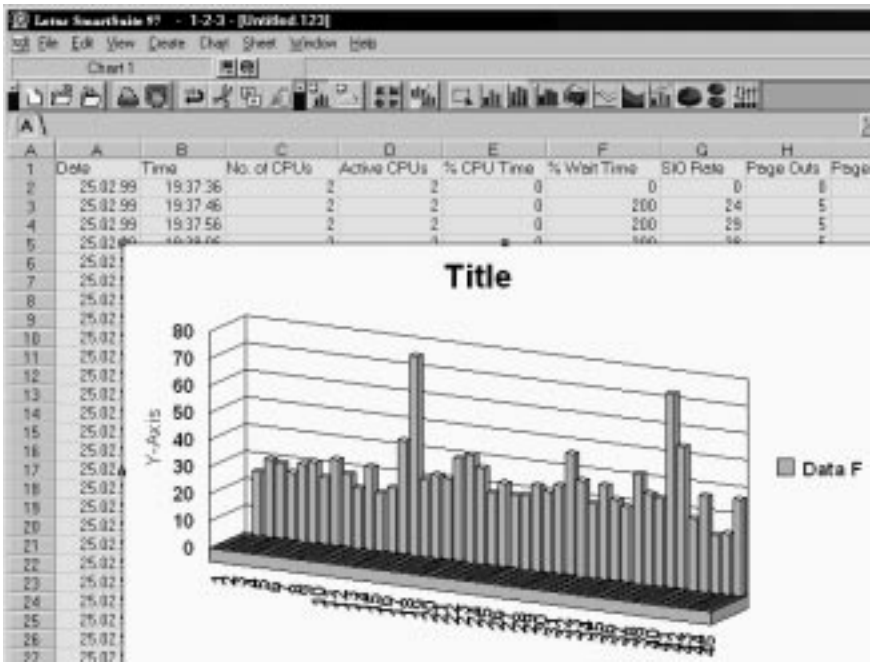


Figure 25. Lotus 1-2-3 Spreadsheet

Copy Files From One VSE Host to Another

This function can be used to directly copy files from one VSE host to another one, even through drag and drop. For example, one VSE library member can be dragged to the library system of another VSE host. This is done without down-loading the file to the client and uploading it to the target host. Instead, the Navigator server establishes a direct TCP/IP connection from the first host to the server running on the second host and transmits the file. VSE phases can even be copied without a LNKEDIT job.

Of course, copying files requires that the Navigator server runs on both hosts. This function is implemented as a server plugin. A client plugin provides a dialog to make the necessary setup.

Job Generation and Tracking

This function provides a mechanism to create compile jobs from skeletons in VSE/ICCF library 2. The ICCF skeletons can be downloaded through another dialog to store them in a PC directory.

For each job that is created and submitted to the host, an object in the *Submitted Jobs* folder is created for tracking the actual status of the job and access related job source and job output. This mechanism is very similar to that implemented in the *VSE/ESA Distributed Workstation Feature (DWF)*. It has some advantages like time event scheduling and working on the original ICCF skeletons, avoiding the more complex syntax of DWF.

By saving the original jobs, it is possible to rerun any previously submitted job at any time without holding the jobs in the VSE/POWER Reader queue.

Other Functions

Display VTOC: This function can be used to retrieve a list of disk addresses from the host. VTOC entries can be displayed for each disk.

Send and Receive Messages from Other Users: VSE Navigator users can exchange messages with this dialog.

Additional functions provide access to the system directory list (SVA), can display the system tasks ordered by partition, view the standard labels, and much more.

Prerequisites

The prototype described in this article has the following prerequisites:

- Client side – JDK 1.1.6 runtime, including Swing classes, installed. For a Windows or OS/2 PC, our prototype needed a minimum of a PII 200 with 64MB RAM.
- Host side – VSE/ESA 2.3 or higher with TCP/IP and Navigator server installed. The server consists of a couple of VSE phases, some configuration members, and a startup job.

Additional Information

The VSE Navigator is not part of any official VSE release. If you want more information about this prototype, just send an e-mail to:

- Ingo Franzki, ifranzki@de.ibm.com, or
- Joerg Schmidbauer, jschmidb@de.ibm.com

We would very much appreciate any suggestions and comments that you send us.

Information Distribution and Exchange with MQSeries

Author

Iris Lee
SWG & TS Integrated Marketing
Information Development
IBM UK Limited
Iris_Lee@uk.ibm.com

IBM MQSeries for VSE/ESA enables your VSE system to exchange information across the enterprise and beyond, with different operating systems on any of the 35-plus platforms supported by MQSeries. The latest version of MQSeries for VSE/ESA Version 2.1, available October 1998, has the same full level of function (known as Level 2) as other MQSeries server products and is available as part of the VSE/ESA V2.3.2 package.

MQSeries frees information held by seemingly incompatible systems so that you can combine and exchange data for new business advantage. MQSeries communicates by sending and receiving data as *messages*. Messages are placed in queues, and the target application takes a message from the queue when it is ready to do so. Network interfaces, communications protocols, and recovery after system problems are all handled by MQSeries.

Publish/Subscribe – Information Where You Want It, When You Want It

In January 1999, MQSeries announced the **Publish/Subscribe** function, a major advance in information distribution. The Publish/Subscribe (pub/sub) function automates the distribution of relevant information to people and applications who have registered interest in a particular topic. It also uncouples applications that provide information from those that wish to receive the information. That is, instead of communicating directly with other applications, an application sends information it wants to share to an MQSeries system with the pub/sub *broker* and lets MQSeries deal with the distribution. The target application doesn't have to know anything about the source of the information it receives.

A provider of information is called a *publisher*. Publishers supply information about a *topic*, without needing to know which applications are interested in the information. A consumer of information is called a *subscriber*. A publisher specifies a topic when it publishes information; a subscriber specifies the topics for which it wishes to receive information. A subscriber then is only sent information for those topics.

The pub/sub broker uses standard MQSeries facilities so, for example, you can get once-only assured delivery, and your messages can be part of a transactional unit-of-work to ensure that messages are delivered to a subscriber only if they are committed by a publisher.

With the Publish/Subscribe function, a VSE/ESA system to be a publisher or a subscriber, provided it is part of an MQSeries network with a route from the queue manager to a platform with the pub/sub broker. “Additional Information” on page 54 describes the platforms on which you can run the broker and how to get it.

Example of a Single Broker Configuration

Figure 26 illustrates a single broker configuration for a stock market organization, where information about several topics is within a single *stream*. A stream is a group of related topics. For simplicity, the figure shows the publishers and subscribers on different platforms, but a system can be both a publisher and a subscriber.

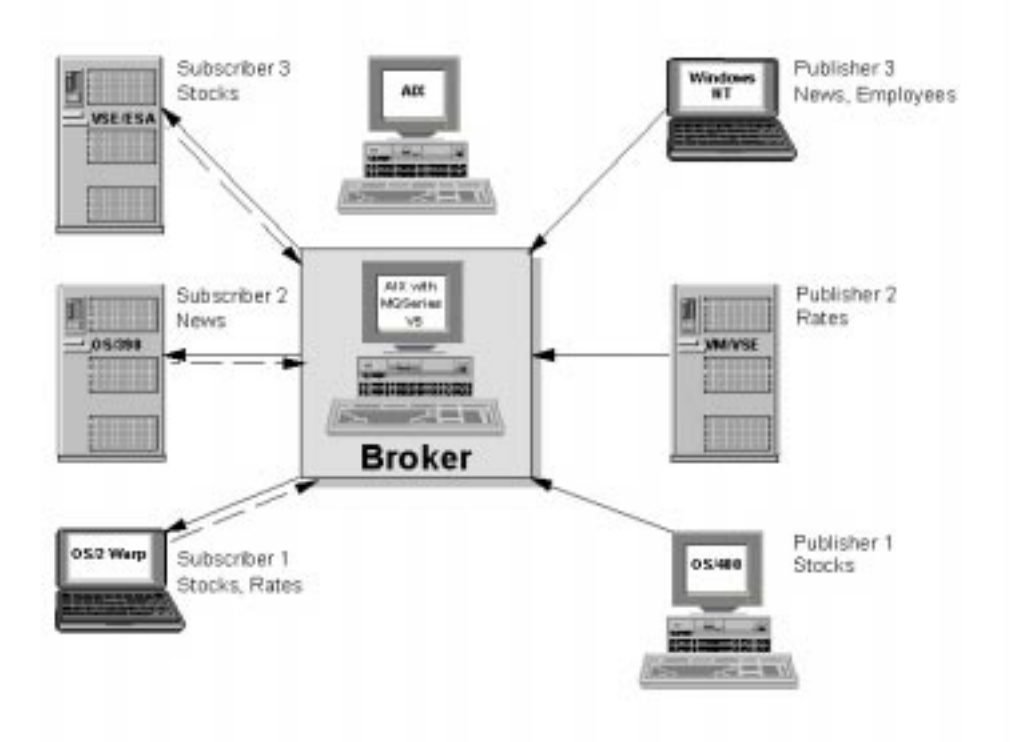


Figure 26. Example of a Single Broker Configuration

In the figure:

- **Publisher 1** publishes information about stock prices using the topic ***Stocks***.
- **Publisher 2** publishes information about currency exchange rates using the topic ***Rates***.
- **Publisher 3** publishes information about financial news items, using the topic of ***News*** and about employee details using the topic ***Employees***.

Three subscribers have registered (shown in the figure by dashed lines) an interest in different topics, so the broker sends them the information they want:

- Subscriber 1 receives the stock prices and exchange rates.
- Subscriber 2 receives the financial news.
- Subscriber 3 receives the stock prices.

No one has registered an interest in employee details, so these are not distributed. Retained publications can be held at the broker until requested by the subscriber.

Subscribers can specify a topic or range of topics, using wildcards, for the information they want.

Publishing and Subscribing

You can have only one broker on each MQSeries queue manager. However, brokers can communicate with other brokers in your MQSeries network, so subscribers can subscribe to one broker and receive messages that were initially published to another broker. A publication is propagated to another broker only if that broker has a subscription to that topic.

Publications are either *state* (retained) publications or *event* publications.

State publications contain information such as the price of stock or the current rate of exchange. When something happens, like a stock price change or a currency rate change, the previous state information is no longer required, since it is superseded by the new information. The subscriber wants to receive the current version of the state information when it starts up and be sent new information whenever the state changes.

Event publications contain information about individual events that occur, such as a trade in some stock or a currency rate reaching a particular level. Each of these events is independent of other events. The subscriber wants to receive information about events as they happen.

Command Messages

Publishers, subscribers, and brokers communicate using simple command messages. Subscribers can register and specify and remove topics. Publishers can register, publish, and remove retained publications. Brokers can exchange subscription registrations and deregistrations and exchange publications and requests to delete publications.

Header Files

As a VSE user, you will need to port the pub/sub header files (**MQRFH**) from a platform that has a broker into your VSE system before you can publish or subscribe. You also will need to specify `convert=yes` on the **SDR** channel of the broker's queue manager to the VSE queue manager to ensure data conversion.

The C-language header files necessary for publish/subscribe applications are:

cmqpsc.h Contains string constants for publish/subscribe messages using the MQRFH header.

cmqc.h Contains elementary data types and some named constants for events and PCF commands.

cmqcfc.h Contains named constants specific to publish/subscribe messages, definitions for PCF structures, and additional named constants for events and PCF commands.

cmqbc.h Contains definitions unique to the MQAI. This file is required only for metatopics and system management functions.

Command Sequences

The following three sequences outline the process of publishing, registering a subscription, and deregistering a subscription.

1. To **publish**:
   a. Prepare the publication.
   b. Put the MQRFH header at the front, and include the `Publish` command and the topic or topics (for example, Stocks).
   c. Send the message using `MQPUT`.
   d. Repeat this sequence as required.
2. To **subscribe**:
   a. Prepare a message with the MQRFH header, and include the `RegSub` command and the topic or topics, using wild cards if you wish.
   b. Send the message using `MQPUT`.

   Having subscribed, you receive messages that other applications publish for as long as you want them.
3. To cancel a subscription (**deregister**):
   a. Prepare a message with the MQRFH header, and include the `DeregSub` command and the topic or topics that you subscribed to, including wild cards.
   b. Send the message using MQPUT.

Additional Information

For more details, see the *MQSeries Publish/Subscribe User's Guide*, GC34-5269-02. Note that publishers and subscribers do not have to be on the same machine as a broker. They can reside anywhere in the MQSeries network, provided that there is a route from their queue manager to the broker. Thus, for example, you could have a publisher on OS/390 and a subscriber on VSE/ESA.

The MQSeries Publish/Subscribe function is available now by installing the MQSeries pub/sub broker function on Version 5.0 of MQSeries for AIX, Sun Solaris, and Windows NT. It will be available shortly on MQSeries for HP-UX.

The Publish/Subscribe function is available at no charge from:
http://www.ibm.com/software/mqseries/txppacs/

COBOL Modernization

Author

Alice Crema
IBM COBOL Family
IBM Santa Teresa Laboratory
crema@us.ibm.com

Are Your COBOL Programs Ready for Year 2000?

When you first embarked on the Year 2000 project, you felt that your programs were in good shape. But when you started to look at your system, did you realize that you weren't really sure about the libraries, let alone the programs? Which programs were really being used? How were the programs built? How do the programs execute? Which applications are dependent on which programs? The questions kept coming, but where are the answers?

Some of you took advantage of the IBM VisualAge® 2000 tools for inventory and assessment, finding and fixing, and testing. Others of you used your own initiatives to understand the questions and find all the answers.

Now That Your COBOL Programs Are Year 2000 Ready ...

Feel good about the completion of your Year 2000 project? Yes, it took some time to figure out where you were, but now you can see more clearly where you *are*.

But what's the next step? Do you think that since you made your COBOL programs Year 2000 ready that you are done with those legacy applications?!?

Think about what was completed during your Year 2000 project. You most likely:

- have a good inventory of your programs and applications.
- know where the date-related logic exists in your programs.
- know how the programs and applications are built.
- know how your databases are structured.
- have better program and application documentation.
- have a better testing methodology.
- have upgraded your system environment software with state-of-the-art products.
- know where future enhancements are needed.
- **have again realized how important and valuable your programmers and legacy applications are to the success of your company!**

Leverage Your COBOL Legacy Assets

The COBOL legacy applications didn't get rewritten to be Year 2000 ready; however, in most cases, they were modified to be Year 2000 ready with either field expansion and/or windowing techniques (manual or automated). Your COBOL legacy applications business logic is understood by your programmers. So, it's time to leverage these important assets.

Now, what do you think of when you hear the words *COBOL legacy applications*?

old ... tired ... headaches ... cobwebs ... nightmares ...
boring black-and-green screen panels ... panic ... etc.

Such negative thoughts about your old COBOL applications, but why? Because of rumors. First came the rumor that the mainframe died, followed by the rumor that COBOL is dead. But the mainframe is growing, and **COBOL is alive!** Remember that COBOL's middle name is ***business***, which means COBOL is real COOL!

Let's Get Positive with COBOL!

Leverage your valuable legacy assets by enhancing your programmer development environment and moving towards e-business. Here's an excerpt from the IBM e-business home page:

> *e-business is the transformation of key business processes through the use of Internet technologies.*

The Web is changing every aspect of our lives, but no area is undergoing as rapid and significant a change as the way businesses operate. As businesses incorporate Internet technology into their core business processes, they start to achieve real business value. Today, companies large and small are using the Web to communicate with their partners, to connect with their back-end data systems, and to transact commerce. This is e-business – where the strength and reliability of traditional information technology meet the Internet.

e-business isn't about re-inventing your business. It's about streamlining your current business processes to improve operating efficiencies, which in turn will strengthen the value you provide to your customers – value that cannot be generated by any other means and value that can give you a serious advantage over your competition.

COBOL Modernization

Legacy code is solid and proven! Your mission-critical applications work! The processing is efficient and good because the applications have been running for years and are well tuned.

It's time to preserve the best of what you've created! Speed up traditional S/390 application development by enhancing the development environment with IBM VisualAge tools. Make it possible for the reuse of legacy assets which will help reduce the development costs and risks when introducing new functions. Leverage legacy skills more effectively by enabling migration to new programming models when appropriate for the business.

Modernization provides the ability to leverage existing software for delivering new business value by:

- Understanding / mining existing business functions, both business and data logic.
- Transforming / re-engineering logic into componentry.
- Providing support for testing the extended capabilities.
- Rapidly deploying new and extended applications into production with minimal disruption.

What Is Legacy Modernization?

Re-engineer your mission-critical legacy applications:

- Legacy enhancements, legacy repair
- "Webify" legacy applications
- Enterprise JavaBean components

The traditional COBOL programmer codes presentation logic, business logic and data access. To "webify" an application is to modify the presentation logic along with maintaining the business logic and data access. Building legacy components involves logic / data introspection and extracting / identifying the business function to componentize.

IBM VisualAge Services offers a roadmap to COBOL modernization:

Migrate – Move to a modern Year 2000-ready base
Master – Increase productivity and prepare for the transition to e-business
Move to the Internet – Help leverage legacy IT assets in a Web world
Maximize – Enable the creation of component-based applications

Let COBOL Legacy Applications Live On!

Ensure that your existing programming assets continue to provide existing business value for Year 2000 and the euro. Continue to leverage the existing development skills. *They are key to your business success*.

IBM will continue to provide tools, components, services, and more as you ***modernize*** your legacy applications!

Additional Information

For more information, please use these Web sites:

International Data Corporation (IDC) article, "Application Development Strategy for the 21st Century"
http://www.ibm.com/software/ad/va2000/

IBM VisualAge 2000
http://www.ibm.com/software/ad/va2000/

IBM COBOL
http://www.ibm.com/software/ad/cobol/

IBM e-business
http://www.ibm.com/e-business/

RPG II to COBOL Translation Service

Author

Jonathan Beit-Aharon
J & C Migrations
migrate@tiac.net

Overview

In the last issue of the *VSE/ESA Newsletter* (Third/Fourth Quarter, 1998), an article about REXX had an item subtitled "Call COBOL, PL/I, or C Programs as Subroutines". This subtitle highlights the limitations of the RPG II compiler on VSE. But there is an easy, economical way to get your RPG II applications to the LE environment and receive all its benefits.

If some of your software assets are RPG II programs, you are maintaining not only their sources but also their development environment and the technical skills. Further, if you are engaged in large-scale repair projects, such as euro or Year 2000 conversions, then you probably already know that fewer tools automate repair of VSE RPG sources and how hard it is to coordinate COBOL and RPG repairs.

J & C Migrations offers an attractive RPG II translation service that is a cost-effective way to maintain your past investment in the development of your applications. The COBOL it generates is readable and LE compliant. It also provides features that will help you train your staff in the use of COBOL.

Economics of Maintenance

Software development is often measured as either Lines of Code (LOC) or Function Points (FP, a language-independent measurement introduced by IBM in 1979). According to Capers Jones of Software Productivity Research, programming one FP takes on average 107 COBOL LOC and 31 hours. Using these numbers, a cost per COBOL LOC can be formulated with a few assumptions as follows:

```
Annual salary and bonus of COBOL programmer .. $80,000

Overhead burden at 35% ..................... $28,000

Monthly burdened compensation ..............  $9,000

Monthly hours devoted to development ......... 134 of 168

Development Cost per Function Point .......... $2082.09

Development Cost per COBOL LOC ..............  $19.46

Defect correction cost (at 2.27%) ............    $.44
```

Even if you are able to spend the $19.90 per working COBOL LOC, for time dependent projects such as euro and Year 2000 enhancements, the duration of the project is not acceptable.

With the assumption that a line of RPG code is equivalent to 3 lines of COBOL, your working RPG LOC can be valued at $59.70 each.

The costs of maintaining applications in a mixture of languages are somewhat hidden. Mismatched skill sets hamper the shifting of personnel and waste time by causing the under-utilization of resources. Language-specific automated repair (mass change) tools differ in their strategies and require additional licenses, coordination, data bridges, and gasket routines – all of which add to the overall complexity of the application, without adding value. Because RPG skills are not readily available when the work needs to be done, for the purpose of maintenance, the hidden costs of having RPG in the code mix can be considered a defect affecting every RPG LOC on every round of maintenance.

The Nature of RPG II

RPG II is the only 3rd Generation Language (3GL) to evolve from a utility program. It is encoded in fixed templates and is not structured into definition and procedure sections. Data items and complex procedures are defined together, often on the same source line.

RPG II tends to be extremely concise. Thus it is possible to write a 20 source-line program which reads a consecutive transaction file, updates an indexed master file with some of its transactions, and prints a report. However, RPG II also tends to be dense and cryptic, and it is hard to maintain cryptic code. These factors have implications both on the FP measurement of RPG LOC and for the costs of maintenance.

RPG II buffers I/O, with the input occurring long before a record is processed. This allows programs to peek ahead and trigger special processing. This also means that input data definitions create working storage for the named areas of file records. The translation generates clear record areas that reflect the file-field correspondence and the related working storage fields.

RPG II program control flow is predefined in a program execution order called the *RPG II Cycle*. Input, calculations, and output are executed in turn as subroutines of the cycle. J & C Migrations provides the cycle in a COBOL copybook and is proud of a unique solution for the management of cycle files without pointers. This solution allows the COBOL code to execute as a single unit, with no need for a separate runtime.

Each RPG II specification has three logic elements – a precondition, a primary purpose, and a post condition. For example, an input precondition is record identification. Its primary function is extraction of a field's data from the I/O buffer, and a post condition can be indicated when the value of the extracted numeric field is negative. The condition states are named "indicators" and are referenced by two-character symbols, shown in predefined locations on each specification template.

COBOL Generated by J & C Migrations

As the initials of **CO**mmon **B**usiness **O**riented **L**anguage suggest, COBOL is indeed the most common 3GL in the world, with commercial quality facilities available on mainframes, minicomputers, UNIX servers and workstations, and PC systems. The current COBOL standard, American National Standard X3.23-1985, was amended in 1989 and 1992. The ANSI and ISO/IEC committees are currently developing the next version of the COBOL standard, with screen I/O, recursion, dynamic data allocation, and Object Orientation among its many features. The COBOL generated by the J & C Migrations translation service is at the latest standard grade.

COBOL allows long and descriptive names, and its Procedure division is structured like English composition, with statements, sentences, and paragraphs. Our translation of RPG II to COBOL does not change cryptic RPG II names into descriptive COBOL names; but wherever possible, it takes advantage of the COBOL ability to be self documenting. Figure 27 illustrates this. In the example, the original RPG II with embedded DL/I is retained as comments in the generated COBOL code. The RPG II reference to the DBSOLD "file" is clarified in the COBOL as a reference to the DBSOLD-RECORD area, which is the target of the INTO phrase.

```
C*          GU        RQDLIDBSOLD                50
C*          'SOLDROOT'QSSA 'SOLDTO  'LOW       EQ
      set f-50 to true
      exec DLI GU using PCB(01)
                  segment (SOLDROOT)
                  where (SOLDTO = LOW)
                  into (DBSOLD-record)
      end-exec
      move dibstat to STCD01
      if dibstat not = '  '
        set t-50 to true
      end-if
C*          STCD01    COMP 'G '                    H2
      set f-H2 to true
      if STCD01 = 'G '
        set t-H2 to true
      end-if
```

Figure 27. Sample Generated Code

The above example also shows how compact RPG II tends to be (three lines to fourteen for COBOL). But the resulting COBOL is much more readable!

COBOL generated by J & C Migrations is not only standard compliant. It also avoids almost all standard elements that are "implementer defined." For example, the standard defined USAGE PACKED-DECIMAL is used rather than COMP-3. This is better documentation and more portable. Also, because COBOL is case insensitive, the translated code highlights names originating in RPG by showing them in uppercase, while generated code is lowercase. This highlighting is a code generation option, and uppercase code generation is available.

The Control and File specifications, plus the U indicators (external switches), create the Identification and Environment divisions of the COBOL program.

All data definitions (which in the RPG II source are scattered throughout the program) are reflected in the COBOL Data Division. Field names that appear on file input specifications appear in both the file section record layout and in the working-storage section.

Clearly named working storage is generated for the logic triggers ("level breaks" and "matching fields"). Edit codes and edit words are also shown with their PICTURE strings in the working-storage section.

Record-level input specifications are reflected under procedure division paragraphs whose names have the prefixes:

- identify- (Record Identification logic),
- sequence- (Record Sequence Checking),
- match- (Matching Fields logic),
- control- (Level break logic), and
- extract- (Extraction from record buffer to working-storage fields).

Calculation specifications are reflected, respectively, between the paragraphs Calculations and Calculations-exit, between Total-Calculations and Total-Calculations-exit, and/or between LR-Total-Calculations and LR-Total-Calculations-exit. Paragraphs are also generated for `BEGSR`, `ENDSR`, and `TAG` operations.

Output specifications create the paragraphs prefixed with out-record- and out-except- (Record level), and the paragraphs format- (Field level).

Array and table initialization, including related file I/O, appear in a COBOL paragraph named load-tables-and-arrays.

Other Benefits

The JCMRPG translator program is a full compiler, and the translation process guarantees cleanly compiled COBOL by performing a COBOL compile. The warnings generated by the two compilers are examined with great care, to ensure the highest quality service. The compile warning lists are provided along with the COBOL sources.

Warnings expose dead code by highlighting statements that cannot be reached and paragraphs that are not referenced. Optimization can be provided by J & C Migrations or carried out by the customer.

Summary

For the purpose of long term maintenance, the hidden costs of having RPG in the code mix can be considered a defect. By performing the RPG to COBOL translation, the investment in application development is maintained, routine maintenance is simplified, and the advantages of the LE environment and future technological advancements become available.

Additional Information

For further information about the services offered by J & C Migrations, go to:
http://www.tiac.net/users/migrate/

CICS Macro Level Source Conversion vs Using a Macro Adapter

Authors

Richard Verville
Jacques Tourigny
VTA Software Inc.
rverville@vtasoft.com

With the new CICS Transaction Server for VSE/ESA, VSE customers are faced with a number of non-support issues. One of these is CICS macro level code programming. The following article covers the most common problems in converting macro level code and why a macro adapter product can be a valid alternative in some circumstances. The first section addresses the problems of macro level code and what you have to look for before you start converting your code. The second section explains how our macro adapter product, **MLI**, resolves the problems.

Common Problems in Converting Macro Level Code

Five years ago, while I was working as technical consultant for CICS in an MVS service bureau, we had clients faced with the macro level problem. CICS 2.1.2 was the last version supporting macro level code, and support from IBM was ending December 1996.

Some clients were already running CICS/ESA 3.3 and 4.1, but others could not move on the new version because of applications containing macro level code. As with CICS TS for VSE/ESA, this code did not work on CICS/ESA.

Sure, you always can rewrite a couple of programs in command level, but what do you do with hundreds and thousands? The first requirement is to have the source of the programs, and that was not always the case for some of our clients.

The very first application we looked at was an insurance package with 530 assembler programs. It turned out that 90% of the problems involved converting macro level code to command level.

Let's look at this from a technical view. For example, coding a program link in macro requires the following:

```
COPY            DFHCSADS
COPY            DFHTCADS
...
DFHPC           TYPE=LINK,PROGRAM=TEST1
```

Converting this example in command level would be:

```
                EXEC CICS LINK PROGRAM('TEST1')
```

Although this looks like an easy one, the macro expansion from the assembly reveals one problem – register usage.

Out of all the programming languages (BAL, COBOL and PL/I), macro level code with Assembler programs is bound to be the most complicated of all.

Programming Differences from Macro to Command

Program Register Usage

In macro level, a simple link to another program ties up 3 registers – R12 for the TCA, R13 for the CSA, and R14 as the link register.

In command level, you'll need to free up Reg0, Reg1, Reg14, and Reg15. You'll also need to have a register for the Exec Interface Block (EIB) and one for Exec Interface Storage (EISTG), totalling 6 registers.

Registers Are Passed in Macro Level, NOT in Command Level

Registers from a macro program that links to another are passed as-is to the next program. Although this is not a good programming practice to assume this, it is used by a couple of application packages I have seen.

Making this work in command level as-is, is not possible. You will need to save the registers somewhere (for example, in the TWA, a Tsqueue, or pass a commarea and restore the registers in the called program).

CWA and TWA

In macro level, using the CWA and the TWA involved adding your variables after the copy statements for DFHCSADS for the CWA and DFHTCADS for the TWA, respectively.

The CWA and the TWA are directly appended in storage after the CSA and the TCA in CICS Version 2 and earlier. This is no longer true in CICS/ESA and CICS TS.

In command level, you must insert an address command in your code and supply a register for each area. You also must insert two DSECT statements in front of your variables for each area. For example, in macro coding:

```
                COPY   DFHTCADS
TWAVAR          DS     CL4
                ...
                COPY   DFHCSADS
CWAVAR          DS     CL4
```

Looks like this in command level:

```
DSECTTWA        DSECT
TWAVAR          DS     CL4

DSECTCWA        DSECT
CWAVAR          DS     CL4
*
                EXEC CICS ADDRESS CWA(R2) TWA(R3)
USING           DSECTCWA,R2
USING           DSECTTWA,R3
```

In command level, you must use the `EXEC CICS ADDRESS` command to get addressability for these zones.

For the CWA, another programming technique in CICS Version 2 and earlier involved coding `EXEC CICS ADDRESS CSA(reg)` and then adding 512 bytes to the returned value. In CICS/ESA and CICS Transaction Server, this is no longer valid.

The returned address from CICS is fetch-protected; and if you try to access it, you will get an abend ASRD. In addition, the CWA and the TWA are no longer appended in storage after the CSA and the TCA, respectively.

Finally, you will no longer be able to gain addressability to the TCT user area by chaining the TCT. You'll need to issue an `EXEC CICS ADDRESS TCTUA(Reg)` and map your DSECT on the returned register value.

DFHBMS TYPE=(IN,OUT),MSETADR=,MAPADR=,

There are macro calls for which there is **no** command level equivalent. The above is one example. This technique involved coding the BMS map in the program itself and then using the mapset address parameter on the BMS calls to display it.

Because `MSETADR` and `MAPADR` have no equivalent in command level, the user must pull the map from the program, make a separate mapset, and finally refer to it with the new mapname.

This is not necessarily complicated. But it is time consuming.

DFHXX CTYPE=LOCATE

This macro call is used to get access to CICS tables like FCT, PPT, PCT, TCT.

Programs containing such calls have to be converted, too. Also note that on certain of the CTYPE macro calls, return upon the call from CICS is at `+4` if the entry is found and at `+0` if not found.

Fortunately, the command level API contains all you need to do the equivalent. You can look at the `EXEC CICS INQUIRE` command. A number of parameters allow scanning each table and retrieving most of the information an application program needs.

VSAM/ISAM Compatibility (CICS 4.1 and CICS TS Related)

If your programs access VSAM/ISAM compatible files, then they're padding 16 bytes on read and write calls for the FIOA DSECT area.

In command level, it is not necessary to pad the 16 bytes; and you must eliminate the FIOA references.

In macro, VSAM/ISAM files are defined unblocked in the CICS V2 FCT. Those files, when used with macro programs, have the FIOA reference.

Also note that you no longer can define these files unblocked with CICS TS. You must use RDO (resource definition online) for VSAM files, and it does not offer this parameter. All files, except for BDAM, on CICS TS are assumed to be blocked.

Performance on Program Links

The biggest surprise from a performance point of view is CPU usage on program links. One of the applications we migrated was linking to function modules that were called (linked) millions of times during the day. When running this application in command level, CPU increased 35%.

We did a small benchmark:

- A BAL program linking 1000 times to another program in macro and then the same programs in command level.
- In command level, the CPU cost for the benchmark was 2.5 times greater than in macro level.

Although we arranged to compensate this with our macro adapter product, there is no easy solution to this problem when converting source code.

Macro Level Interpreter (MLI)

In this section, I'll describe how our macro adapter product handles the above problems. The objective of the product is to run macro level programs under CICS/ESA or CICS Transaction Server *without the need of any programming changes by the customer*. Macro programs under CICS TS run with MLI as if they were running CICS V2 native.

Program Register Usage

In MLI, we recreate the CICS V2 environment for certain CICS control blocks, one of which is the TCA. We save the registers in the TCA and pass them to the called program.

When the macro program gains control, Reg 12 points to the MLI TCA allocated at task startup, Reg 13 points to the MLI CSA, and Reg 14 points to the entry point of the program. All other registers are those of the calling program.

CWA, TCA and TCT User Area

CWA: At MLI startup (normally at PLT), MLI getmains an area to contain the CSA, CWA (size is based on the SIT definition), and the CSA optional feature list – all in V2 format. We copy the current content of the system CWA to the MLI CWA.

Programs handled by MLI then see the MLI CWA. The MLI CWA is never copied back into the system CWA. This can create the problem where you could have non-macro programs that need to share CWA data with macro programs. MLI resolves the problem by handling the command level program as well.

Because MLI also supports mixed mode programs (command level programs with macro calls or control block references), it sees every EXEC CICS command prior to its execution.

When an EXEC CICS ADDRESS CWA is issued, MLI passes back the MLI CWA address to the application. In this way, all programs handled by MLI see the MLI CWA. MLI also intercepts EXEC CICS ADDRESS CSA calls and returns the MLI CSA address. This is necessary because programs may address the CSA just to gain access to the CWA.

TWA: Just before the task starts, MLI getmains an area big enough to contain the TCA in V2 format (X'190' bytes), plus whatever size is specified for the TWA.

Register 12 points to the MLI USER TCA when the program gets control. The TWA is X'100' from Reg 12 like in CICS V2.

When a program is given control, MLI copies the system TWA into the MLI TWA. Upon return of the program, MLI copies the MLI TWA into the system TWA. In this way, both the MLI TWA and the system TWA are always in sync. This is neces-

sary, since a task could have a mix of MLI-handled macro programs and non-handled programs. MLI also intercepts EXEC CICS ADDRESS TWA calls on MLI-handled programs and returns the MLI TWA address.

TCT user area: MLI points to the system TCTUA. There is no double copy.

In CICS TS, the SIT parameter `TCTUALOC` sets the TCT user area location either above or below the 16MB line. Because macro level programs were designed to run in 24-bit, the customer must have the TCTUA below the line.

DFHBMS TYPE=(IN,OUT),MSETADR=,MAPNAME=

With MLI , we developed an internal BMS function to support this macro call without requiring any changes from the customer.

DFHXX CTYPE=LOCATE

MLI supports these calls as-is for application-related tables such as the PPT, PCT, FCT, TCT and the DCT.

In order to keep MLI storage utilization to its minimum, these tables are built dynamically as the resources are used by the macro programs. In this way, there is no double copies of full tables in storage.

There are, however, applications that issue a CTYPE locate (for example, on the FCT to verify the status of files before any file operations). If the table is not already pre-built, the application does not see its files.

To alleviate this problem, MLI can be customized using online menus to pre-build table entries. The user can tell MLI to add only selected entries or to build all the entries available in the CICS region.

All these tables can be viewed online with the MLI CORE transaction, which makes it easy when debugging applications.

VSAM/ISAM Compatibility

MLI totally supports these types of files. Because there is no way to tell CICS TS about unblocked files and MLI has to know when to pass an FIOA to the application, we have made a MLI online menu in which the customer can tell MLI the ddnames of VSAM/ISAM files.

For each of these ddnames, MLI sets the unblocked bit on in its FCT. The FIOA can then be returned to the program, leaving the application unchanged when running on CICS TS.

CICS SNT Under Transaction Server

There is no internal security with CICS TS. Because some macro applications chain the SNT from the TCTTE for internal security reasons, MLI also builds a V2 SNT table based on the information returned from CICS.

With most external security packages, you can still supply a 3-byte Operator ID, commonly used by applications. When a MLI-handled program starts, we issue an `EXEC CICS ASSIGN opid` command and set the TCTTEOI field with the CICS returned information. In this way, applications can still behave as they did on Version 2.

Performance

The most frequent question asked about MLI is: *How much overhead is involved running macro applications with MLI?*

This varies on a case-by-case basis. There is minimal overhead incurred by MLI and much more overhead incurred by certain command level calls.

The section that follows explains what is involved with MLI. The second explains what we did in MLI to reduce the command level overhead on program links and storage requests.

For MLI overhead, there are 2 aspects – CPU and storage

1. **CPU** – As odd as this may sound, there is really not much work involved in translating macro calls into command. When a macro call is issued, the program falls into the MLI CSA pointers that contain each MLI subroutines to handle the calls. The request bytes are examined, and the appropriate command call is issued by MLI to CICS. When return is given to MLI, we set the TCA fields with the response and return control to the macro program. The CICS service is still really done by CICS, not MLI. We just act as interface between the macro program and CICS.

   For mixed mode programs (command level program with macro calls), there is a little bit more work involved, as we see all command calls from the application. We have to look for EXEC CICS ADDRESS commands and handle conditions. We estimated overhead of that work to be 3 to 5%.

2. **Storage** – For each macro program, not mixed mode, we attach a stub in front of the program, once at program load time. This is required by CICS to treat the program as a command level. The stub size is the same as the command level stub, 40 bytes for Assembler and 72 bytes for COBOL. For PL/I, a stub is always present and is reused by MLI.

   At MLI initialization, we create V2 control blocks and tables based on user requirements. The total size varies, depending on user setup. We supply a transaction which shows how much storage is used by each tables.

   At transaction start, we getmain an area large enough to hold the TCA in V2 format and the TWA specified on the PCT definition. For each getmain requests made by the macro program, MLI prefixes the storage with 12 to 20 bytes of header information replicating the V2 storage accounting area and other chaining requirements.

Program links and storage requests

1. **Program links** – We were fortunate to have had an insurance package with millions of program links as our first major application to work with MLI. Because the command level overhead on program links were so high, the customer was rethinking his move on CICS/ESA.

   This problem got us to develop the direct-link feature in MLI, a user adjustable (driven by online menus) table where the customer codes the names of program that are highly used.

   When MLI activates, it looks in this menu. For each program found, it issues an EXEC CICS LOAD PROGRAM HOLD, thus locking its location. On subsequent program links or XCTL to this program, MLI branches directly in the program instead of issuing an EXEC CICS LINK PROGRAM call.

This has the same effect, as often seen in packages, where a couple of programs are loaded with the HOLD parameter, the addresses are stored in the CWA, and the application branches directly in the routines instead. The only difference is that the direct-link feature lets the customer adjust his/her application based on needs, and this does not require any programming changes.

We have seen customers running MLI with the direct-link on CICS/ESA with less CPU than that required running native on CICS V2. The drawback is the storage below the line tied up by the programs, much like resident programs on CICS V2.

2. **Storage requests** – Another high user of CPU is the storage getmains/freemains. It is common in macro to getmain/freemain a lot of storage (for example, on file operations or TIOA manipulation).

   With MLI, we've came up with an internal storage chain built on a per-transaction basis. This is called the *storage reserved option*.

   When turned on, MLI keeps on an internal storage chain a certain amount of Kbytes (based on user parameters) for an individual task. When a freemain request is issued, MLI does not release the storage to CICS but keeps it to satisfy a future getmain request. Another parameter controls how the getmain request will be satisfied (for example, so we don't tie up a 100 bytes for a 10 bytes request). At task termination, all storage is released to CICS.

   Applications issuing of lot of these calls will see their CPU consumption reduce 10 to 30%.

Conclusion

I've tried to cover most common macro related problems in this article. The decision to convert or to use a macro adapter product has to be based on the effort required to convert source, assuming it is available. In addition, it is not necessarily on how much programs you have but how complicated is it to convert them.

Originally we thought that most of our customers would have been those without source code. It's not. 80% of our MLI clients have application packages that will eventually be replaced in a couple of years. Thus it is just cheaper to run a macro adapter than to convert source code.

Additional Information

If anyone has questions/comments on any the above, we'll be glad to answer them. You can e-mail us directly at **support@vtasoft.com** or call us at **450 448-3346**. Our Web site is **http://www.vtasoft.com/**.

MLI a registered trademark of VTA Software Inc. MLI is in North America distributed by:

MacKinney Systems
2740 S. Glenstone, Suite 103
Springfield, Missouri 65804

Phone: (417) 882-8012
Fax: (417) 882-7569
e-mail: sales@mackinney.com
Web: http://www.mackinney.com

Creating REXX Functions

Author

Patrick Talley
Senior Systems Programmer
Altec Industries, Inc.
pataIley@altec.com

Overview

In this article, I will describe how to create functions in REXX and in HLASM. To illustrate this, I have written the same function in both languages. This function is part of a utility application I wrote to assist me on a project to eliminate Shareoption(4) files. I needed to be able to distinguish COBOL keywords from variable names so I could properly identify the files being opened in a program.

Besides creating a function, there were other options for doing this – like hard coding several IF statements into a REXX procedure or using a large SELECT routine. I discarded these ideas since there are over 300 reserved COBOL keywords, and I wanted a way to do this identification efficiently.

My goal in the main REXX program was to pass a character string (which could be a COBOL/VSE reserved word or variable) and get a simple "yes" or "no" answer after performing a binary search. This is exactly what can be accomplished using a REXX function.

I originally was only going the write this function in REXX, but I ran into some problems getting it to work the way I wanted it to. I came back to it after I got an HLASM version to work. Thus I now can show the same function in REXX and HLASM for your benefit, as well as my own.

Required REXX Keywords

Writing a function in REXX is fairly easy, since there are only three keywords needed to create a skeleton for a function: **PROCEDURE**, **ARG** and **RETURN**.

The function is called by using it in place of a REXX expression. This can appear in different formats, but the simplest is:

```
answer=name(var1,var2).
```

The variables passed to the function can be any REXX expression, including a function call. However, you can not pass an array in the form of a stem variable. Thus `answer=name(stem.)` is invalid.

Internal and External Functions

There are two types of REXX functions you can define – internal and external. The main difference is where the function is defined. An internal function is in the same PROC member as the main calling routine. An external function is contained in a separate PROC member.

To start the definition of a internal function, use the PROCEDURE keyword in the format:

```
name: PROCEDURE
```

If you want to share data between the function and the calling routine, there is an optional parameter, `EXPOSE varname`, for the PROCEDURE keyword. This allows the function to use and update these variables, and should be used with caution.

The PROCEDURE keyword is omitted for an external function. Instead, the function begins with a `/* comment */` line like a regular REXX procedure.

The function name is the name of the PROC the function is contained within. Since there is no PROCEDURE keyword, you have to pass all of the data in the function call.

Receiving Values

To receive values that are passed to the function use, the `ARG` keyword in the format:

```
ARG var1, var2
```

Ending a Function

To end the function, use the `RETURN` keyword in the format:

```
RETURN(expression)
```

The expression is an optional parameter, where the result is passed back to the calling routine. The expression is typically a variable name, but it can be any REXX expression such as a calculation or a function call. The RETURN keyword does not have to be physically at the end of the function. It can be placed anywhere that the function can logically return to the calling routine, in multiple locations if necessary.

Recursive Calls

One of the nice things about writing a function in REXX is that it allows for recursive calls. The binary search routine I used is a recursive routine, so it lent itself to being implemented in REXX. However, this does require a slightly different mind set from the typical programming logic, especially if you are not accustomed to structured programming. My function calls itself for each table entry to be examined for the keyword passed to it.

Logic Issues

Some of the logic issues that I ran into were:

- How to keep all of the function logic out of the main program.
- Where to determine the address of the next table entry to be checked.
- Which data values needed to be passed forward to the next function call.

The solution to the first issue was to make my function *two* functions. My main program first called **rxcbrsv**, which initialized my array with COBOL/VSE reserved words and variables for determining the first entry to be checked in the array. Then this function called the second-level function **rxrsvrd**, which performed a binary search of the table. See Figure 28 on page 75.

Assembler Program RXCOBRSV

Loading the table was one of the problems I had with implementing the function in REXX. Since it is an interpretive language, the logic for loading the array is executed each time the main program calls the function. I could get around this by loading the array in the main program and using the EXPOSE option so it will be available to the function, but that defeats the idea of having a self-contained function and keeping that logic out of the calling program. This is what prompted me to first implement this function in HLASM (Figure 29 on page 76).

Remaining Issues

The other two issues are really related and probably could be resolved in several different ways. What I decided on was to calculate the size of the subset to be searched, the offset to the midpoint of that subset and the address of the entry to be checked in the calling function, and pass them forward along with the keyword being searched for.

I wish I could say I came to this decision after careful thought and planning, but in reality I arrived at this through trial and error. I consider this to be another advantage of writing a function in REXX. With the "say" and "trace" commands the debugging process is quick and easy.

Unless you are more comfortable with Assembler code than REXX, writing a function in HLASM is more difficult and time-consuming than it is for REXX. The good new is that the interface for getting the input data from the REXX program and returning the result is simple.

Finding an Example: The first step that I would recommend in creating a function in Assembler is to try and find an example that is close to what you are trying to write. I started by looking through the examples directory of the VSE Collection CD-ROM and found a sample REXX function in **RXPROG52**. This basically provided me the logic for handling the input and output routines for the function. I uploaded this sample program and used it as a skeleton program for my REXX function. I then used **IGY8RWRD.Z** to create a table of the reserved words for the lookup. I was not able to find any sample code for performing a binary search, so I had to develop that logic on my own.

Understanding Linkage Logic: The next step was to refer to the *VSE/REXX Reference* manual to understand the linkage logic used in the sample program. The REXX linkage data areas are created through three macros – **ARXEFPL**, **ARXARGTB** and **ARXEVALB**.

ARXEFPL Defines the external function parameter list (EFPL). There are two key addresses in the EFPL, EFPLARG which points to the argument list (the input data) and EFPLEVAL which points to the evaluation block (the result to be passed back).

```
EFPLARG                         DS A
EFPLEVAL                        DS A
```

ARXARGTB Defines the argument list. The list is an array of addresses and lengths ending with two fullwords of high-values marking the end of the array.

```
ARGTABLE_ARGSTRING_PTR          DS  A
ARGTABLE_ARGSTRING_LENGTH       DS  F
```

ARXEVALB Defines the evaluation block. `EVALBLOCK_EVSIZE` is the size of this data area in double words. `EVALBLOCK_EVLEN` is the length of the result the function returns. It is initialized to `X'80000000'`. `EVALBLOCK_EVDATA` is the area for returning the result to the calling REXX procedure.

The size of the EVALBLOCK_EVDATA is EVALBLOCK_EVSIZE * 8 - 16. This defaults to 256 bytes. If you need to return more than 256 bytes of data, you can use the ARXRLT routine to acquire a larger area.

Before returning back to the calling routine you will need to load the length of the data being returned to EVALBLOCK_EVLEN and of course move the result into EVALBLOC_EVDATA.

```
EVALBLOCK_EVSIZE                DS  F
EVALBLOCK_EVLEN                 DS  F
EVALBLOCK_EVDATA                DS  C
```

Register Usage: Finally, the last thing you need to be concerned about is the register usage. Upon entry into the function:

- Register 0 points to the environment block (defined by ARXINIT) of the calling program,
- Register 1 points to the EFPL,
- Register 13 points to the register save area,
- Register 14 is the return address and
- Register 15 points to the entry point in the function.

Once you have the logic for handling input and output for REXX completed, all that is left is regular Assembler programming. In my case, I got a good start because I had already figured out the basic logic from writing the function in REXX. I did put in a little extra effort on defining the table so I could easily update or replace it.

Incorporating an External Function into a Function Package

The performance of an external function that is called frequently can be improved by incorporating it into a *function package*. This is because an external function is reloaded into the partition each time it is called, whereas a function package is loaded only once.

Creating a function package only requires a little additional work. First, you need to pick a name for the function package. There are two default names that can be used: **ARXFLOC** and **ARXFUSER**. If you choose to use another name, you will need to create a customized ARXPARMS module or call the ARXINIT routine to make it available to your REXX program. Also note that some of the software packages on your system already may use the default function name, so take this into consideration when you pick a name.

The next step is to assemble the HLASM function and catalog the object module that will be link-edited into the function package. This also can be done with a REXX function if you have a REXX compiler.

The final step is to create the function package directory. There is no macro to assist in the creation of the directory, but there is a macro, **ARXFPDIR.A** in PRD1.BASE, that supplies the field names and layout.

The directory has two sections, a header and the entries. Three of the fields in the header are constants:

1. **FPCKDIR_ID** is set to ARXFPACK,
2. **FPCKDIR_HEADER_LEN** is the length of the header and will always be set to X'00000018', and
3. **FPCKDIR_ENTRY_LENGTH** is the length of the directory entry and will always be set to X'00000020'.

The fourth field, **FPCKDIR_FUNCTIONS**, contains the number of directory entries in the function package.

```
FPCKDIR_ID                      DC CL8
FPCKDIR_HEADER_LEN              DC F
FPCKDIR_FUNCTIONS               DC F
FPCKDIR_ENTRY_LENGTH            DC F
```

The function package will have one directory entry for each function or subroutine in the function package. Each directory entry has three fields to be set:

1. **FPCKDIR_FUNCNAME** is the name of the function pointed to for the entry,
2. **FPCKDIR_FUNCADDR** is the address of the entry point for the function, and
3. **FPCKDIR_SYSNAME** is the name of the entry point for the function.

```
FPCKDIR_FUNCNAME                DC CL8
FPCKDIR_FUNCADDR                DC A
FPCKDIR_SYSNAME                 DC CL8
```

FPCKDIR_FUNCADDR and FPCKDIR_SYSNAME serve the same purpose, and only one has to be specified. Please note that there are a few reserved fields in the header. For the complete layout of the directory and the entries that I have omitted from this article, refer to **ARXFPDIR.A** in PRD1.BASE or to the manual, *VSE/REXX Reference*. Figure 30 on page 78 shows my sample code for ARXFLOCL.

Performance

To illustrate the performance issues with the different ways you can define functions, I ran four tests. In each test, I called the function 33,903 times using the reserved word array along with three extra words to verify the logic of the function.

- Case one used the HLASM function.
- Case two used the HLASM function incorporated into a function package.
- Case three used an external REXX function.
- Case four used an internal REXX function and loaded the reserved word array once in the main program and passed it the function via the EXPOSE option.

As you can see below, the external REXX function has the worst overall performance. There is a significant I/O overhead associated with the HLASM function and the external REXX function. This comes from loading the function each time the function is called. The I/O overhead is eliminated for the HLASM function by it into a function package.

Table 1. Performance Comparison

Test Case	CPU Seconds	I/Os Performed
HLASM function	75.652	135,789
HLASM function package	27.224	218
External REXX function	2,627.15	1,186,721
Internal REXX function	480.169	152

Additional Information

When choosing which type of function you want to use, you should consider the following:

1. Does the function require any language specific logic?
2. How often will the function be called from the main program?
3. How many different REXX programs will call the function?

If your function requires specific logic or capabilities of REXX or HLASM, that will dictate the use of that language for the function. If the function will be called several thousand times from the same main program, you should rule out an external function. If only a few REXX programs need to call the function, you should consider an internal REXX function. If the function will be used heavily, the best choice would be a HLASM function incorporated into a function package.

Please note that a zipped file with my code is available via the VSE/ESA home page. To get this file, go to:

http://www.ibm.com/s390/vse/vsehtmls/s390ftp.htm

Look for **rexxfunc.zip**.

```
* $$ JOB JNM=RXCBRSV,DISP=D,CLASS=T,PRI=4
* $$ LST DISP=H,CLASS=P
// JOB RXCBRSV
// EXEC LIBR
ACCESS S=OEM.TEST
CATALOG RXCBRSV.PROC      DATA=NO REPLACE=YES
arg key_word
  /*****************************************************************/
  /* routine to test reserved routine                              */
  /*****************************************************************/
  rsvrd_word.0=339
  rsvrd_word.1='ACCEPT              '
  rsvrd_word.2='ACCESS              '

...

  rsvrd_word.338='ZEROES              '
  rsvrd_word.339='ZEROS               '
chk_index = ((rsvrd_word.0 +1) % 2)
search_size = rsvrd_word.0 - ((rsvrd_word.0 + 1) % 2)
mid_point=((search_size + 1) % 2)
return(rxrsvrd(key_word,chk_index,search_size,mid_point))
rxrsvrd: procedure expose rsvrd_word.
arg key_word,chk_index,search_size,mid_point
ANS='NO  '
search_size_n = search_size - mid_point
mid_point_n=trunc((search_size_n + 1) / 2)
if key_word = rsvrd_word.chk_index then do
  ANS='YES '
  return(ANS)
end
if search_size > 0 then do
   select
     when key_word > rsvrd_word.chk_index then do
       chk_index = chk_index + mid_point
       return(rxrsvrd(key_word,chk_index,search_size_n,mid_point_n))
     end
     when key_word < rsvrd_word.chk_index then do
       chk_index = chk_index - mid_point
       return(rxrsvrd(key_word,chk_index,search_size_n,mid_point_n))
     end
     otherwise do
     ANS='YES '
     return(ANS)
     end
   end
end
return(ANS)
/+
/&
* $$ EOJ
```

Figure 28. RXCBSRV

```
* $$ JOB JNM=RXCOBRSV,DISP=D,CLASS=T
* $$ LST DISP=D,CLASS=P
// JOB RXCOBRSV ASSEMBLE A REXX ASSMBLER FUNCTION
// OPTION CATAL
 PHASE RXCOBRSV
// LIBDEF PHASE,CATALOG=OEM.TEST
// EXEC ASMA90,SIZE=(ASMA90,300K),PARM='EXIT(LIBEXIT(EDECKXIT))'
          REGEQU
          CSECT RXCOBRSV
          USING *,R15
          SAVE  (R14,R12)
          BALR  R10,R0
          USING *,R10                    BASE REG 10
          LR    R2,R1                     LOAD REG 2 FROM 1
          USING EFPL,R2                  POINT TO PARMLIST
          L     R3,EFPLARG               POINTER TO ARGUMENT LIST
          L     R9,EFPLEVAL              SAVE EVALUATION BLOCK POINTER
          USING ARGTABLE_ENTRY,R3
          L     R4,ARGTABLE_ARGSTRING_PTR POINT TO FIRST ARGUMENT
          L     R5,ARGTABLE_ARGSTRING_LENGTH LOAD ARGUMENT LENGTH
          CLRRTN KEYWORD
*  R0 = TABLE ENTRY SIZE
*  R1 = TABLE SEARCH SIZE
*  R2 = 2*TABLE ENTRY SIZE
*  R3 = POINTER INTO TABLE
*  R4 = PREVIOUS TABLE SEARCH SIZE
          LM    R0,R1,TABLE
          CR    R5,R0
          BH    NOTFND
          BCTR  R5,R0                   SUBTRACT 1 FOR EX
          EX    R5,EXMVC
          LA    R3,TABLES
          LR    R4,R1
TABHIGH   A     R1,=X'00000001'
          SRL   R1,1          CALCULATE MIDPOINT(R1/2)
          LR    R11,R1
          MH    R11,TABLE+2
          AR    R3,R11        GET POSITION INTO TABLE
          CH    R1,=X'0000'    IS OFFSET LESS THAN OR EQUAL TO TWO
          BNH   ENDCHK        YES, THEN CHECK LAST TWO ENTRIES
*                             TABLE ENTRIES?
```

Figure 29 (Part 1 of 3). RXCOBRSV

```
TABLOW    SR    R4,R1          SET UP R4 AND
          LR    R1,R4                 R1 FOR NEXT PASS
          CLC   0(L'TABLES,R3),KEYWORD
          BE    TABFND
          BL    TABHIGH
          A     R1,=X'00000001'
          SRL   R1,1          CALCULATE MIDPOINT(R1/2)
          LR    R11,R1
          MH    R11,TABLE+2
          SR    R3,R11        GET POSITION INTO TABLE
          CH    R1,=X'0000'    IS OFFSET LESS THAN OR EQUAL TO TWO
          BNH   ENDCHK        YES, THEN CHECK LAST TWO ENTRIES
          B     TABLOW
ENDCHK    SR    R3,R0         SHIFT BACK
          CLC   0(L'TABLES,R3),KEYWORD
          BE    TABFND
          A     R3,TABLE      SHIFT TO NEXT ENTRY IN TABLE
LSTCHK    CLC   0(L'TABLES,R3),KEYWORD
          BE    TABFND
NOTFND    LA    R7,NO                  SET R15 TO ONE
          B     SAVRTN
TABFND    LA    R7,YES                    SET R15 TO ZERO
SAVRTN    L     R8,0(R9)                   POINT TO EVALBLOCK
          USING EVALBLOCK,R8
          LA    R6,X'00000004'          SET REPLY LENGTH
          ST    R6,EVALBLOCK_EVLEN      STORE REPLY LENGTH IN EVALBLOCK
          MVC   EVALBLOCK_EVDATA(4),0(R7)
          SR    R15,R15                  SET R15 TO ZERO
          ST    R15,16(R13)              SAVE R15 IN SAVE AREA
          RETURN (R14,R12)               RETURN TO CALLING FUNCTION
EXMVC     MVC    KEYWORD(1),0(R4)
KEYWORD   DS     CL20
ADDTAB    DC     A(TABLES)
YES       DC     C'YES '
NO        DC     C'NO  '
TABLE     DC     A(L'TABLES)
ENTRIES   DC     A((TABLEE-TABLES)/L'TABLES)
TABSIZE   DC     A(TABLEE-TABLES)
          LTORG
```

Figure 29 (Part 2 of 3). RXCOBRSV

```
TABLES    DC     C'ACCEPT              '
          DC     C'ACCESS              '

...

          DC     C'ZEROES              '
          DC     C'ZEROS               '
TABLEE    EQU    *
          DS    0F
          ARXARGTB
          ARXEFPL
          ARXEVALB
          END
/*
// EXEC LNKEDT,PARM='AMODE=31,RMODE=ANY'
/*
/&
* $$ EOJ
```

Figure 29 (Part 3 of 3). RXCOBRSV

```
* $$ JOB JNM=ARXFLOC,DISP=D,CLASS=T
* $$ LST DISP=D,CLASS=P
// JOB ARXFLOC ASSEMBLE A REXX FUNCTION PACKAGE
// OPTION CATAL
 PHASE ARXFLOC
// LIBDEF *,SEARCH=TEST.BASE,CATALOG=OEM.TEST
// EXEC ASMA90,SIZE=(ASMA90,300K),PARM='EXIT(LIBEXIT(EDECKXIT))'
ARXFLOC   CSECT ARXFLOC        REXX FUNCTION PACKAGE DIRECTORY
FPCKDIR_ID          DC  CL8'ARXFPACK' FPCKDIR CHARACTER ID
FPCKDIR_HEADER_LEN  DC A(FPCKDIR_HEADER_END-FPCKDIR_ID)
FPCKDIR_FUNCTIONS   DC A((FPCKDIR_END-FPCKDIR_FUNCNAME)/(FPCKDIR_ENTRY_*
               END-FPCKDIR_FUNCNAME))
                    DC  F'0'     RESERVED
FPCKDIR_ENTRY_LENGTH DC A(FPCKDIR_ENTRY_END-FPCKDIR_FUNCNAME)
FPCKDIR_HEADER_END        EQU *
FPCKDIR_FUNCNAME    DC CL8'RXCOBRSV'   NAME OF FUNCTION OR SUBROUTINE
FPCKDIR_FUNCADDR          DC  V(RXCOBRSV)  ADDRESS OF THE ENTRY POINT
*                                    of the package code
                          DC  F'0'     RESERVED
FPCKDIR_SYSNAME           DC  CL8' '    NAME OF THE ENTRY POINT
*                                    corresponding to package code
FPCKDIR_SYSDD             DC  CL8' '   RESERVED              @VSEDH
FPCKDIR_ENTRY_END         EQU *
FPCKDIR_END               EQU *
          END
/*
 INCLUDE RXCOBRSV
// EXEC LNKEDT,PARM='AMODE=31,RMODE=ANY'
/*
/&
* $$ EOJ
```

Figure 30. ARXFLOC

What's New in DB2 Server for VSE & VM Version 6?

Author

Hernán Sarazola
Senior Database Administrator
Ediguay S.A. Data Processing Services
and Consulting
Montevideo, Uruguay
zarasola@netgate.com.uy

Notes:

1. Hernán has a Web home page with database information:
   http://www.netgate.com.uy/~zarasola/index.htm
2. This article refers to *Application Servers* and *Application Requesters* in a DB2 Server for VSE & VM environment. For definitions of these terms, refer to page 85 in Rich Smrcina's article, “Now Showing – Data from VSE on a Web Browser Near You.”

TCP/IP Support for DB2 Server for VM

TCP/IP stands for *Transmission Control Protocol/Internet Protocol*. TCP/IP is a set of rules that allow computers with different hardware and software to communicate across a network.

TCP/IP is a two-layered program. The higher layer, *Transmission Control Protocol*, manages the assembling of a message or file into smaller packets that are transmitted over the net and received by a TCP layer that reassembles the packets into the original message. The lower layer, *Internet Protocol*, handles the address part of each packet so that it gets to the right destination. Each gateway computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they'll be reassembled at the destination.

TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service by another computer (a server) in the network. TCP/IP communication is primarily point-to-point, meaning each communication is from one point (or host computer) in the network to another point or host computer.

For the DB2 Server for VM Application Server, TCP/IP support is invoked at system initialization time. If TCP/IP for VM is available, the server will make use of it. The Application Server must be able to determine what port number to listen on for connections.

Interaction with the Application Server is simple. Two new initialization parameters, some messages, and some new command output are involved:

1. The first initialization parameter is **TCPPORT**. Valid values are integers in the range of **0** to **65535**.

   Specifying TCPPORT in single user mode is ignored. If 0 is specified, then no TCP/IP initialization will be attempted.

If TCPPORT is not specified, TCP/IP initialization uses the **ETC SERVICES** file to determine the port number that should be used.

2. The second initialization parameter is **TCPPORTR**. Valid values are integers in the range of **0** to **65535**.

   Specifying TCPPORTR in single user mode is ignored. If 0 is specified, then no resync port will be established. DRDA2 connections via TCP/IP are rejected.

   If TCPPORTR is not specified, then TCP/P initialization uses the ETC SERVICES file to determine the port number that should be used.

3. New messages are displayed at initialization to inform the operator if TCP/IP support was successfully enabled or not. The service name and port number are displayed.
4. Existing operator commands will be enhanced to support the TCP/IP functions.

With two new initialization parameters and three sets of values for each, there are nine combinations of initialization scenarios. Combined with some of the existing initialization parameters, knowing what happens during initialization becomes quite complex.

If a DB2 Server for VM Application Requester wants to access a remote database using TCP/IP, an entry in the communications directory, **COMDIR**, must be set up. Currently, the COMDIR is used to provide SNA network information to access a remote database. The COMDIR must be set up to provide the host and service names the requester will use on the connection.

Stored Procedures for VM and VSE

A stored procedure (sometimes called *database procedure* or *triggered procedure*) is a precompiled program that is stored at the server site and is known to the server and is callable by a local or remote Application Requester (AR).

Some advantages of stored procedures are that:

- They provide a greater degree of data independence (concealing a variety of system-specific or database-specific details from the user).
- They can be shared by many clients.
- You can hide the details of the database design from client applications.
- You have more flexibility for security:
  - A given user might be authorized to invoke a given procedure but not to operate directly on the data accessed by that procedure.
  - Stored procedures can be used to keep the processing of sensitive data within the database's local environment, thus eliminating the possibility of malicious manipulation of data that flows through the network.
- Optimization can be done at the time the stored procedure is created instead of at run time.
- You can move most of the SQL™ statements and data processing logic into a stored procedure that is executed locally at the Application Server (AS) location. This means that most of the work can be done locally, reducing network traffic.

Functional Description

In DB2 Server for VSE & VM, all stored procedures are "fenced." This is necessary to ensure that a stored procedure does not:

- Inadvertently use storage that is allocated to the database manager.
- Monopolize processing in the database machine or partition (the database would hang).
- Effect execution on the database machine if it terminates abnormally.

A fenced implementation is achieved through the use of *stored procedure servers*, as follows:

1. In VM, a stored procedure runs as a "private resource" in a separate virtual machine that is local to the DB2 Server for VSE & VM server.
2. In VSE, a stored procedure runs in a separate static or dynamic partition.

When a stored procedure is running, the storage procedure server in which it is executing is dedicated to it. Since no other storage procedure can execute in that server until the current one finishes, it is not possible to execute multiple stored procedures concurrently with a single stored procedure server. However, multiple stored procedures can execute concurrently if multiple stored procedures servers are defined.

From the Application Server, DB2 Server for VSE & VM 6.1 accepts CALLs to stored procedures from any Application Requester using SQLDS and/or DRDA protocol and can provide result sets for DRDA requesters. ARs on VSE & VM can use SQLDS protocol to CALL a stored procedure. VSE & VM ARs do not support CALLs when using DRDA protocol. Since results sets are implemented through DRDA, results sets are not available to VSE & VM ARs.

DRDA RUOW Application Requester Support for CICS/VSE

Distributed relational data function may be classified by the constraints placed upon the application:

- **R**emote **U**nit of **W**ork – At execution time, the remote unit of work supports multiple SQL requests within an application program, but all requests within an LUW access a single remote database.
- **D**istributed **U**nit of **W**ork – This also supports multiple requests within the application program, with each request accessing a single database. However, the different requests within an LUW may access different databases.
- **D**istributed **R**equest – This allows the above, plus the ability for each request to access multiple databases.

The Distributed Relational Database Architecture (DRDA) support being added to DB2 Server for VSE 6.1 will consist of RUW AR support for online CICS/VSE application programs. This support will provide CICS/VSE online application programs with the ability to execute SQL statements to access and manipulate data managed by any remote Application Server that implements the DRDA architecture. The SQL statements in these application programs can be *static*, *dynamic* and *extended dynamic*, even if the target system does not support extended dynamic statements.

The CICS/VSE online application program will only be able to access one Application Server (remote or local) per LUW. A COMMIT or ROLLBACK must be issued to terminate the LUW before an attempt is made to connect to another Application Server.

When the SQLGLOB parameter SYNCPOINT 2 is specified, a CICS transaction can, within the same LUW, update a remote Application Server and another CICS resource which participates in two-phase commit. (**Note:** VSAM does not participate in two-phase commit.)

RDS Above 16MB Line

The RDS component has been made AMODE(31) compatible to allow it to be loaded above the 16MB line. This frees up approximately 2MB of storage below the line (3MB if DRDA is enabled), allowing to have more users defined, for example.

If your machine or partition is defined > 16MB, RDS will be loaded above the line if there is room. However, if you also run with AMODE(24) but your machine or partition size is > 16MB, message ARI0021E will be displayed; and the database will not start.

Data Restore Feature Incremental Backup

The Data Restore **BACKUP** function backs up the whole database. However for many Applications Servers, only a small part of the database is modified frequently.

The **INCREMENTAL BACKUP** function only backs up the parts of the database which have been modified since the last full backup. Since an incremental backup contains all pages that have been modified since the associated FULL backup was taken, incremental archives increase in size until a new full backup is taken.

Users can RELOAD or RESTORE from incremental backups.

Generally, incremental archives will be most useful for large databases of which a small percentage of the data is updated.

The **FULL** parameter has been added to the DB2 Server for VSE & VM ARCHIVE operator command to allow the archive to be used as a starting point for taking incremental archives with Data Restore.

Unicode, Euro Currency Symbol and Code Pages

Support has been added for:

- CCSID conversion from Unicode to host CCSIDs (E-International, E-English, Katakana, Kanji, Japanese-Katakana, Japanese-Kanji, Korean, Simplified Chinese and Traditional Chinese).
- Euro code page versions of English, International, UK-English, French, German and Italian. DB2 also provides some programs to convert from a non-euro to an euro-compatible version of the same language (for example, from German to E-German).
- New Latvian/Lithuanian, Estonian, Ukranian, Vietnamese and Laotian code pages.

New Features

QMF/VM, QMF/VSE and QMF for Windows are now available as features of DB2 Server for VSE & VM. These used to be program products on their own but are now only available as features of DB2 Server for VSE & VM.

As an ordering convenience, DB2/Connect also is being made available as a feature.

NLS Installation Changes for Base and Features

- All customers receive all languages for DB2 Server for VSE & VM.
- An additional install tape ("Help Text") is shipped to all customers.
- The installation process has been modified to install any language during installation.
- A language also can be changed post-installation.
- All features (except QMF) now ship all languages.
- Data Restore, Drop Capture already have combined languages.
- Control Center/VM is only available in English.
- Control Center/VSE has all languages combined on the base tape.
- REXX SQL has all languages combined on the base tape.

Additional Information

For more information about the DB2 Server for VSE & VM, access this Web site:
http://www.ibm.com/software/data/db2/vse-vm/

Now Showing – Data from VSE on a Web Browser Near You

Author

Rich Smrcina
Grede Foundries, Inc.
Milwaukee, Wisconsin
rsmrcina@grede.com

Overview

This article demonstrates how we at Grede Foundries (Milwaukee, Wisconsin) are able to directly access our VSE-based data from a Web browser. There are many ways to accomplish this task, and many vendors will claim to be able to do it with their products. Some succeed; some fail miserably.

A number of IBM products all work together to make this long-standing dream a reality: **Domino Go Webserver for Windows NT**, **DB2/Connect for NT**, **eNetwork™ Personal Communications for NT**, **Net.Data for NT**, and **DB2 Server for VSE**.

This capability has been a long-standing requirement for both Grede management and IT support. Since our existing software suite is prohibitive in that area, I made the suggestion to try DB2 for VSE. We knew that DB2 could accomplish all of the tasks that were required, but the idea of implementing another database was a very hard sell. In addition, we are in the middle of the Y2K crunch and also are implementing a couple of ERP (Enterprise Resource Planning) applications. There is precious little time to dedicate to a new database.

Although this article focuses on Web serving, most of the configuration tasks also are applicable to using ODBC[4] applications like Microsoft Access, Microsoft Excel, Lotus Approach, Visual Basic or Crystal Reports to access DB2 from a workstation.

The Project

The goal of the project was quite simple – to demonstrate that data residing on DB2 could be displayed and manipulated with a Web front end and that PC-based applications also could access DB2. We already had DB2 for NT, and as a pilot project, the goal was accomplished. But there are just a couple more hoops to jump through when going after DB2 for VSE.

Thus when the opportunity arose to test DB2 for VSE, we readily agreed. I had some previous DB2 experience, so I took on the task of installing and configuring the test database. Our current database allows arrays of objects (or an array of a group of objects) to appear inside of a database "record." Most of those arrays are broken out into separate tables. Except for that, the design of the test database mirrors our existing production database. We're currently doing proof-of-concept testing. Database (re-)design will be saved for when DB2 for VSE is actually implemented.

[4] **O**pen **D**ata**b**ase **C**onnectivity. See the article, “Using ODBC and SQL to Access VSE Data” on page 103.

The project has had a number of phases:

- Installing DB2 on VSE and taking an initial stab at database design.
- Configuring all of the communications pieces to allow machines on the LAN access to DB2.
- Loading our operational data and providing the users with access to Web applications or showing them how to use workstation programs to access DB2.

Installing DB2 on VSE and Configuring Communications

Installing DB2 on VSE was quite easy. The installation manual takes you through all of the appropriate steps. One step in the install process that was critical to the entire project was installing the DRDA code.

Distributed Relational Database Architecture (DRDA) is software and a set of rules that governs how DB2 communicates across platforms. This includes, for example, how workstations communicate with VSE and (in the case of DB2 Version 6.1) how VSE communicates with other platforms.

As indicated in the article by Hernán Sarazola (“What's New in DB2 Server for VSE & VM Version 6?” on page 79), two concepts are necessary to understand when working in a DRDA environment: **Application Server** and **Application Requester**.

An Application Requester (AR) is any DB2 client that makes an SQL request to a DB2 server on another platform. An Application Server (AS) is any DB2 server that can receive requests from a DB2 client on another platform.

Clients usually communicate to mainframe DB2 from Windows 95, Windows NT, AIX, or OS/2. AR and AS are typically used when discussing communications between a mainframe and DB2 on another platform using the DRDA protocol. If a Windows 95 DB2 client wished to access a Windows NT DB2 server, it would not use DRDA. It would typically use ODBC.

Mainframe DB2 does not always have to be the server either. Starting with DB2/VSE Version 6.1, VSE can be a client (although client functionality is restricted to CICS, not batch). This means that if NT, AIX or OS/2 DB2 are databases out on the LAN, a CICS application can be written to access them just as if they were local to VSE.

DRDA is implemented natively by the mainframe DB2 server products (like DB2 Server for VM & VSE). Windows, AIX and OS/2 require additional software (called **DB2/Connect**) to implement DRDA.

Communications for DB2/Connect is handled by **Communications Server** (for NT or AIX) or **eNetwork Personal Communications** (PComm for NT). At Grede, we installed DB2/Connect and PComm on the same Windows NT machine. We call this machine the *DB2 Gateway*. PComm is used to implement APPC on the gateway machine. DB2 for VSE currently only supports APPC as a communications protocol.

Configuring APPC for use with DB2 was quite a challenge; but when it was done, the results were amazing. Performance was actually better than was expected. SNA performance tips are available that helped considerably.

Figure 31 is a simple representation of our configuration.

Figure 31. Basic Configuration

On the mainframe, we use the Open Systems Adapter-2 to provide the SNA communications for the DB2 Gateway. VTAM™ definitions are involved, and the part of the SWNET book related to the DB2 Gateway is in Figure 32. Note that DB2LU is the independent LU that DB2/Connect uses to communicate with VSE.

```
WEBSERVE PU    ADDR=01,ANS=CONT,DISCNT=NO,
               IDBLK=05D,IDNUM=00001,DYNLU=YES,MAXDATA=1929,MAXOUT=7,
               MAXPATH=1,MODETAB=IESINCLM,PACING=8,PUTYPE=2,VPACING=8
PATH001  PATH  GRPNM=OSAGRP,CALL=INOUT,DIALNO=0104400000020053
WEBLU01  LU    LOCADDR=0,DLOGMOD=LU62PS
DB2LU    LU    LOCADDR=0,DLOGMOD=IBMRDB
```

Figure 32. DB2/Connect Definitions for VTAM

All communications from DB2/Connect to VSE is routed through CICS. The Application Server is implemented as a CICS transaction. Multiple transactions can be used, depending upon security or performance concerns. When a request is made through DB2/Connect, the communications link initiates a transaction (usually called AXE), which executes the SQL on VSE and returns the response or the result set back to DB2/Connect. DB2/Connect then routes it to the workstation user.

CICS sees DB2/Connect as another interconnected system. The definitions are just like an ISC connection (Figure 33 on page 87).

```
CEDA  View
    Connection     : DB2A
    Group          : DB2CONN
   CONNECTION IDENTIFIERS
    Netname        : DB2LU
    INDsys         :
   REMOTE ATTRIBUTES
    REMOTESystem   :
    REMOTEName     :
   CONNECTION PROPERTIES
    ACcessmethod   : Vtam              Vtam | IRc | INdirect
    Protocol       : Appc              Appc | Lu61
    SInglesess     : No                No | Yes
    Datastream     : User              User | 3270 | SCs | STrfield | Lms
    RECordformat   : U                 U | Vb
   OPERATIONAL PROPERTIES
    AUtoconnect    : Yes               No | Yes | All
    INService      : Yes               Yes | No
  SECURITY
    SEcurityname   :
    ATtachsec      : Identify          Local | Identify | Verify
    Bindpassword   :                   PASSWORD NOT SPECIFIED

CEDA  View
    Sessions       : DB2A
    Group          : DB2CONN
   SESSION IDENTIFIERS
    Connection     : DB2A
    SESSName       :
    NETnameq       :
    MOdename       : IBMRDB
   SESSION PROPERTIES
    Protocol       : Appc              Appc | Lu61
    MAximum        : 00016 , 00016     0-32767
    RECEIVEPfx     :
    RECEIVECount   : No                No | 1-999
    SENDPfx        :
    SENDCount      : No                No | 1-999
    SENDSize       : 16384             1-30720
    RECEIVESize    : 16384             1-30720
   OPERATOR DEFAULTS
    OPERId         :
    OPERPriority   : 000               0-255
    OPERRsl        : 0                 0-24,...
    OPERSecurity   : 1                 1-64,...
    USERId         :
   SESSION USAGES
    Transaction    :
    SESSPriority   : 000               0-255
   OPERATIONAL PROPERTIES
    Autoconnect    : Yes               No | Yes | All
    INservice      :                   No | Yes
    Buildchain     : Yes               Yes | No
    USERArealen    : 000               0-255
    IOarealen      : 00000 , 00000     0-32767
    RELreq         : No                No | Yes
    Discreq        : No                No | Yes
    NEPclass       : 000               0-255
   RECOVERY
    RECOvoption    : Sysdefault        Sysdefault | None
```

Figure 33. DB2/Connect Definitions for CICS

The Personal Communications setup also was involved and required some assistance from DB2/Connect support and PComm support. PComm requires the definition of a PU and LU6.2 sessions. Screen shots of some of the more important definitions are in Figure 34 on page 88 through Figure 42 on page 95

Figure 34 defines the PComm Control Point and Local Node ID, which matches values used in the PU definition.

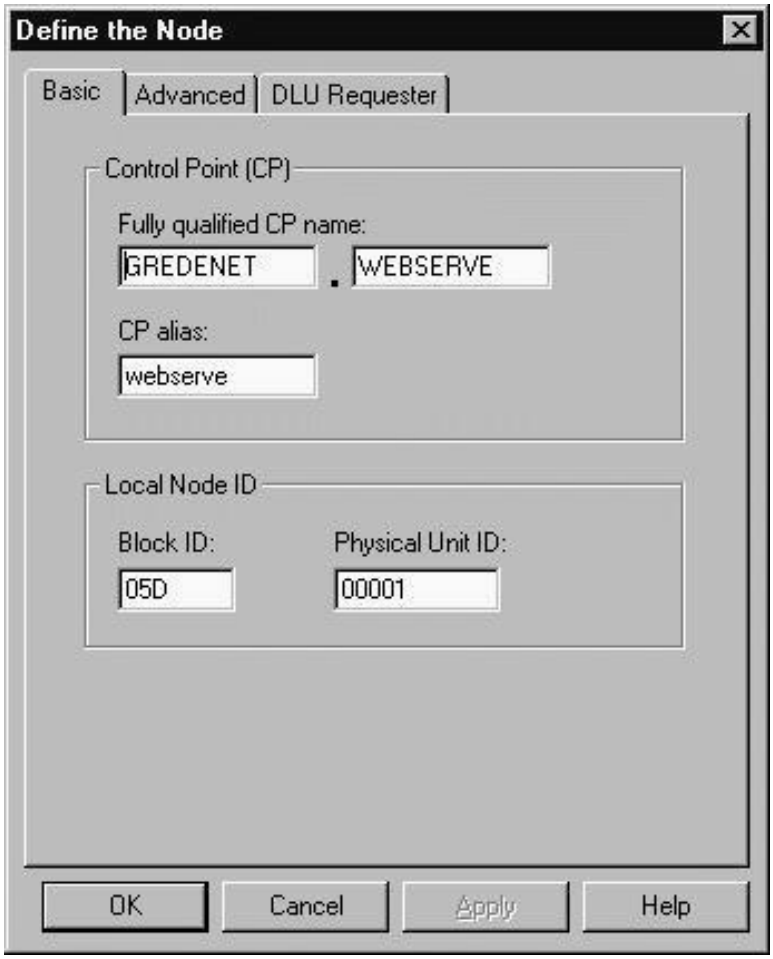


Figure 34. Control Point and Local Node ID Definitions

Figure 35 sets some LAN parameters (namely the remote MAC address, which is the MAC address of one of our OSA-2 ports).

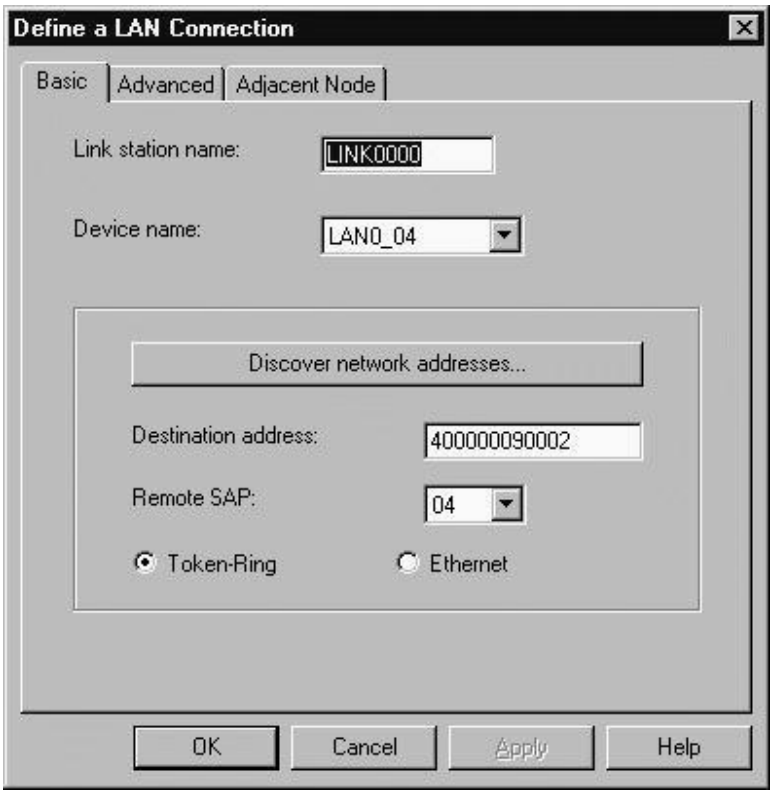


Figure 35. Remote MAC Address Definitions

Figure 36 shows part of the Adjacent Node information. The adjacent CP name is the SSCPNAME that is defined in the VTAM startup book, `ATCSTR00`.

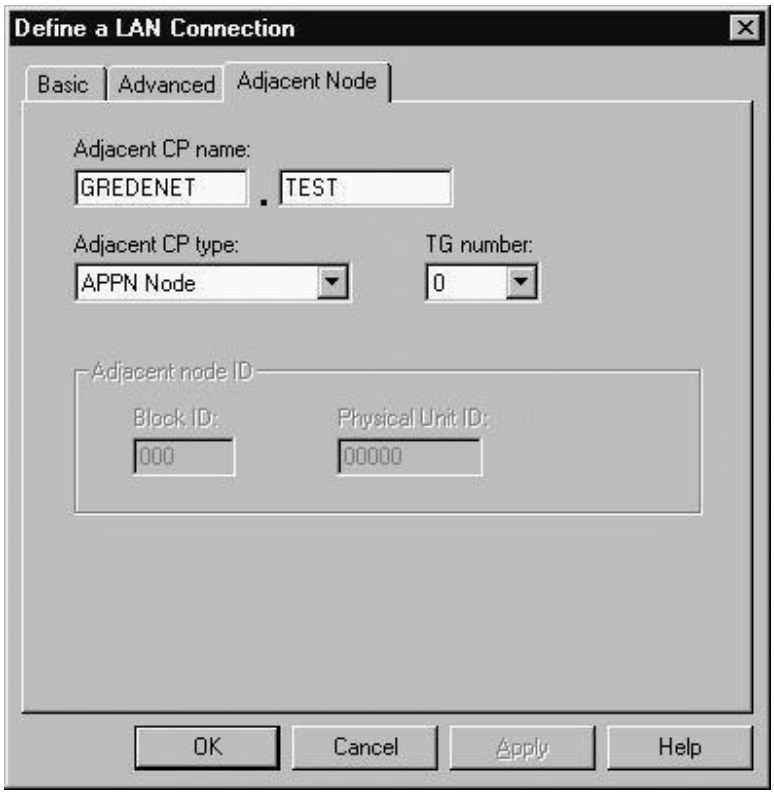


Figure 36. Adjacent Node Definitions

Figure 37 shows the LU6.2 definitions. The partner LU name is the CICS that all DB2 DRDA communications flows through. The fully-qualified CP Name is the same as on the previous screen.

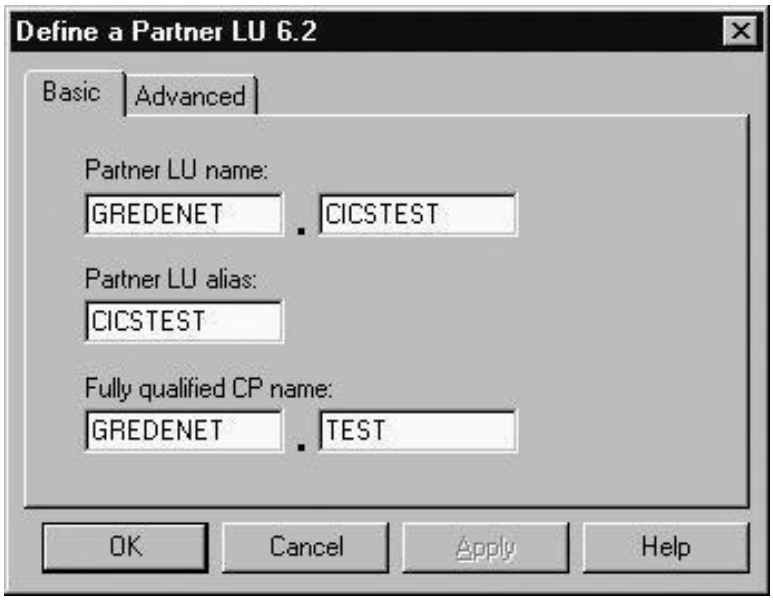


Figure 37. LU6.2 Definitions

Figure 38 shows the PComm LU6.2 definition. The name and alias matches the LU name used in the SWNET book and CICS.

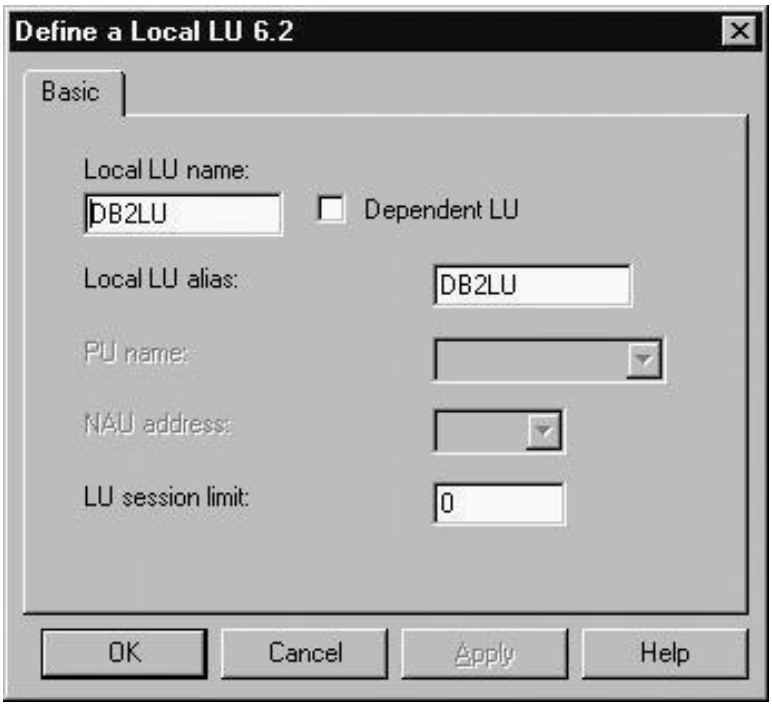


Figure 38. PComm LU6.2 Definitions

Since DB2/Connect uses CPI-C API's, a CPI destination must be configured. This was one of the stumbling blocks for us. The Symbolic Destination Name must be 8 characters long in PComm, but all of the samples written for Communications Server used names with 7 characters.

The other values in Figure 39 match related host values. The TP name is for the transaction that is used to process DRDA requests. This transaction name is defined to CICS in hexadecimal.

Figure 39. CPI-C Side Definitions

Figure 40 shows the DB2 Client Configuration Assistant (CCA) dialog for defining the link between DB2/Connect and VSE.

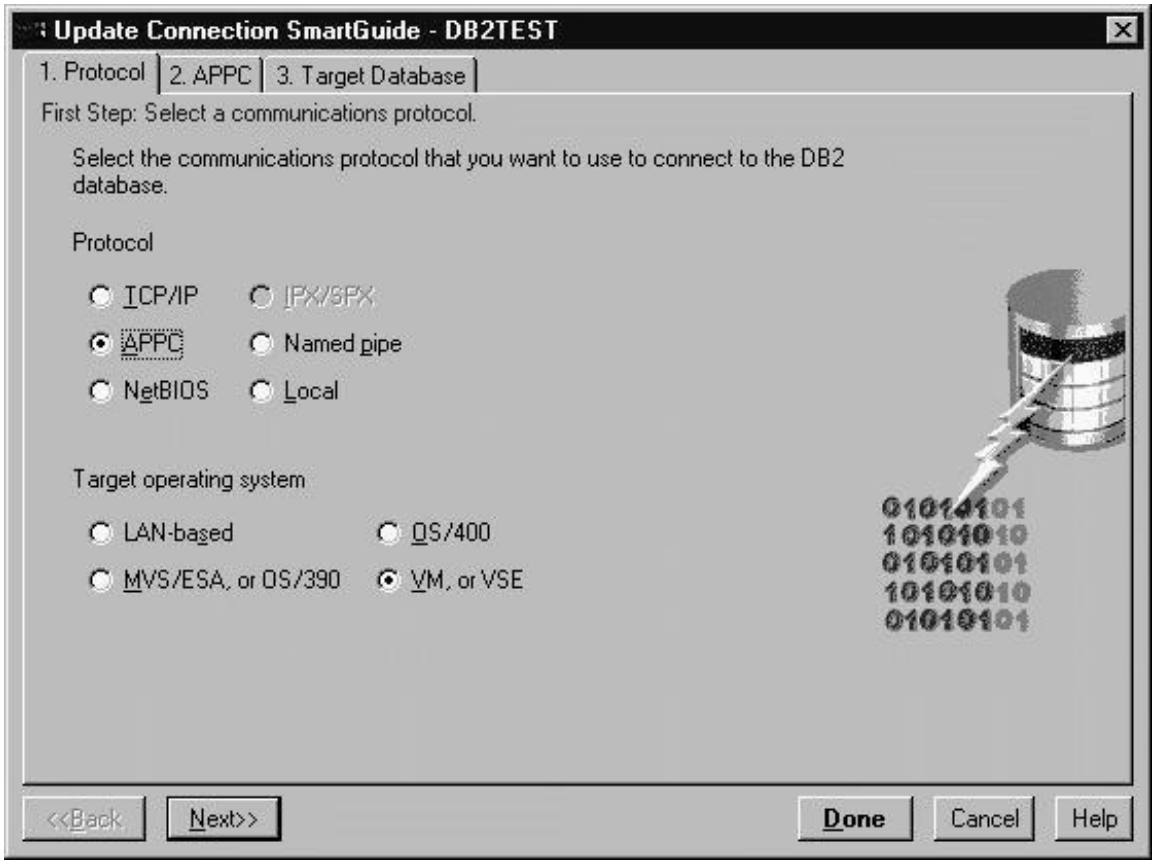


Figure 40. Defining the Link Between DB2/Connect and VSE

The APPC tab in the CCA (Figure 41) has the definition of the CPI-C destination.

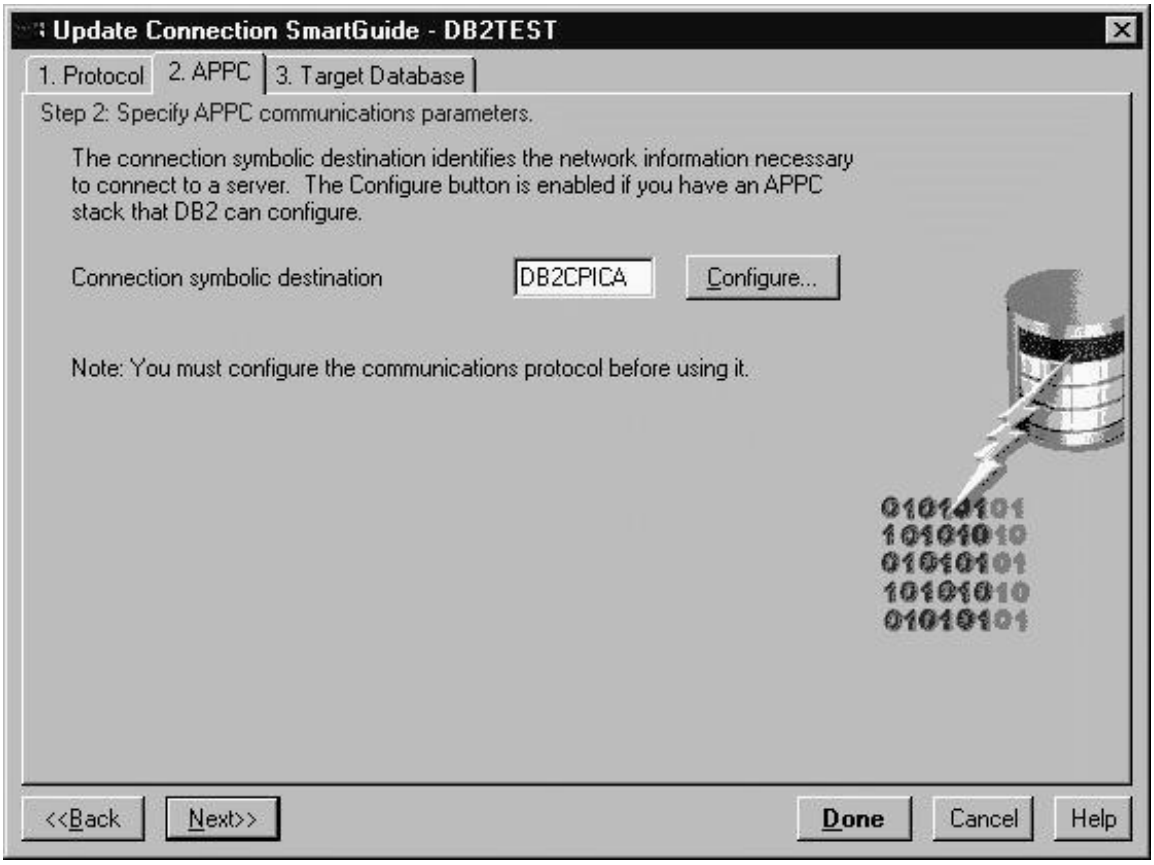


Figure 41. Defining the CPI-C Destination

The last tab in the CCA is for telling DB2/Connect the real name of the database on VSE (Figure 42 on page 95).

Figure 42. Defining the Data Base Name

Loading Operational Data

With the initial database layout completed and tables designed to reasonably mimic our existing database, the next task was loading the tables. The database product used for our production database can unload its data to a flat file for reading by another program. The reformatting of the data to load into DB2 was accomplished using a combination of DFSORT/VSE and VSE/REXX.

DFSORT/VSE has an amazingly powerful `OUTREC` facility that can change the way data read into DFSORT is written. Fields can be moved and written with an edit mask, literals can be inserted, and data also can be written with various date formatting masks. There is even a lookup and replace command set that can replace input values when the data is written.

This facility was extremely useful when dates on the production database contained "special" values that are not valid dates. DB2 flagged these special dates as invalid and did not load the data. With DFSORT/VSE, these values were replaced with valid values on output, and DB2 then loaded them as valid dates. They retained their "special" meaning, just with different values.

When more intense decision-making logic was required in the conversion process, VSE/REXX was used to reformat the data. The conversion and load time varied greatly depending upon the size of the file, as one would expect. The largest single file from our current database – which loaded 1.3 million rows into three different DB2 tables – took about 15 hours to convert with REXX and load into DB2.

Serving Up DB2 Information

The more our users become familiar with PC applications and Web browsers, the more they want to be able to work with them. With the Web browser as the front end and a Web server as the application server that retrieves information from our database, we have successfully merged centralized, decentralized, and network computing at Grede. The "dumb" front end which used to be the 3270 is now the Web browser. It has all of the graphical user-friendly components that have been missing in the 3270 interface. The applications can be served up to the browser by the Web server. In some cases, they even can execute on the Web server (except for Java which generally executes on the client). The data resides on the host, where it has been (or should have been) all along.

The writing of applications that execute on the Web server and that generate HTML displayed on a Web browser usually is done through CGI (Common Gateway Interface) programming. CGI programs are typically written in C or Perl. They generate HTML by "writing" it to the Web browser just as if they were writing to a screen. This often is accomplished in C by using the `printf()` function.

A solution brought to market by IBM a couple of years ago is *Net.Data*. This product can best be described as a programming language. It has most of the components of any high level language (variable assignment, conditional processing, arithmetic processing, looping and an impressive function library). It is designed specifically for use on a Web server and merges programming with HTML generation. The terms "macro" and "program" when used with Net.Data are synonymous.

Net.Data is available for Windows NT, AIX, OS/2, Linux, OS/400®, OS/390, and various flavors of UNIX. Net.Data also will work with a number of Web servers. On Windows NT, for example, it works with Domino Go Webserver, Netscape's Web Server, and Microsoft's Internet Information Server.

Net.Data will run as a CGI on all of the supported Web Servers; but for performance reasons, it is much better to use a feature included with the product called *Live Connection*. When accessing a database, a great deal of time is spent in getting a connection to the database. Live Connection will always keep an open database connection, which can save a considerable amount of time and make your Web pages run much faster. Depending upon how many concurrent users could be accessing Net.Data, you can define how many connections Net.Data should open at once. All of that is tailorable through Net.Data's configuration files.

For the purpose of our project, Domino Go Webserver and Net.Data are both installed on the DB2 Gateway machine. The Web Server's configuration files need to be modified to map a Net.Data URL to the Net.Data executable, so that Net.Data gets control and can execute the program. If Live Connection is desired, it is started as another process and can also be set up as an NT Service.

A typical URL to execute a Net.Data program looks like this:

```
http://is0053/cgi-bin/db2www/program.mac/entrypoint
```

where `is0053` is the internal name of the Web server, `/cgi-bin/db2www` is the path on the Web server that invokes Net.Data, `program.mac` is the name of the Net.Data program, and `entrypoint` is the point in the Net.Data program where execution begins

A Net.Data program can have as many entry points as needed. I have some programs with two entry points:

1. One is for data entry that presents a Web page with entry fields, normally used to fill in search criteria for a query.
2. Another entry point generates a Web page based upon parameters passed in the URL.

In this way, I can write a program that will accept criteria to be displayed and also call the same program from a different program and pass those criteria through the URL. I use this method to link Net.Data programs together.

Net.Data programs can be written using a simple text editor. I usually use Notepad, although anything that can save a text file would work just fine.

Figure 43 on page 98 contains a sample Net.Data program. The program is rather typical. It has a `DEFINE` section that sets up certain values, namely the name of the database. The `%INCLUDE` statement (**1**) copies the contents of the file specified into the program (much like a COBOL copybook). The included file can contain executable Net.Data statements if needed.

In this case, the file contains standard Web page header and trailer HTML that is stored in Net.Data variables (Figure 44 on page 99). Displaying the standard header or footer is as simple as using the variable name in an HTML section of the program. A file like this is used in all programs so that all of the generated Web pages have a similar look.

Net.Data programs rely heavily on curly braces and percent signs to denote block boundaries. Language element blocks are framed by `{` and `%}`, and Net.Data language statements are within the block. For readability, the opening brace, the statements, and the closing percent sign-brace are on separate lines. Variables are prefixed by a dollar sign and enclosed in parenthesis.

All of the punctuation seemed overwhelming at first; but as I became familiar with Net.Data, it occurred to me that it was all necessary so that Net.Data could distinguish its own language elements from the HTML that is coded into the program.

```
%DEFINE {
DATABASE="DB2TEST"
 title="Pattern List"
 SHOWSQL="no"
%}
%INCLUDE "patthdft.inc"                    1

%FUNCTION(DTW_SQL) getlist() {
  SELECT a.pattern_num, a.plant, b.description
     FROM GREDE.PATT_MASTER_CUST a, GREDE.PATTERN_MASTER b
     WHERE a.CUST_NUM='$(custnum)'         4
       and a.PLANT=b.PLANT
       and a.PATTERN_NUM=b.PATTERN_NUM
     ORDER BY plant ASC, pattern_num ASC
  %MESSAGE{                                5
        100: {
             <font color="red">There are no patterns for Customer Number $(custnum).</font>
             %} : continue
    default: "<br>getlist: Return code $(RETURN_CODE) from SELECT" : continue
  %}
  %REPORT{                                 6
    <table border="0" cellspacing="3" width=60%" align="center">
    <caption><b>Pattern List for Customer $(custnum)</b></caption>
    <tr>
    <th align="left">Plant</th>
    <th align="left">Pattern Number</th>
    <th align="left">Pattern Description</th>
    </tr>
    %ROW{                                  7
      <tr>
      <td>$(V_Plant)</td>
      <td><A href="/cgi-bin/db2www/pattern_detail.mac/report?plant_num=$(V_Plant)&pattnum=$(V_Pattern_Num)"
      onmouseover="window.status='See Pattern Master Detail for Pattern Number $(V_Pattern_Num)'; return true"
      onmouseout="window.status='Pattern List for Customer $(custnum)'; return true">
      $(V_Pattern_Num)</a></td>
      <td>$(V_Description)</td>
      </tr>
      %}
    </table>
    <br>Return to
    <A href="/cgi-bin/db2www/pattern_detail.mac/input"
    onmouseover="window.status='Return to Pattern Detail selection page'; return true"
    onmouseout="window.status='DB2 Mainframe Pattern Master Detail display'; return true">
    Pattern Detail</a> selection page.
    %}
%}

%HTML(report){                             2
$(stdheader)
@getlist()                                 3
$(returntomenu)
<hr>
$(stdtrailer2)
</body>
</html>
%}
```

Figure 43. Sample Net.Data Program

Toward the bottom of the program is the entry point. It is an HTML section called `report` (**2**). An HTML section is used when HTML needs to be sent to the browser. HTML is sent to the browser in the sequence that it is encountered in the

program. Thus the beginning of the Web page is generated at the start of the report section. The very first statement in report invokes the standard header variable. The HTML for the standard header is shown in Figure 44 on page 99.

```
stdheader={ <html><head><title>$(title)</title></head>
            <body bgcolor="white">
            <table>
            <tr><td rowspan=5><img src="/images/gredeg.jpg" height=91 width=51 align=middle></td></tr>
            <tr><td><font face="Arial" size=6>Grede Foundries</font></td></tr>
            <tr><td><font face="Arial" size=4>$(title)</font></td></tr>
            <tr><td><font face="Arial" size=2>@DTW_rDATE() @DTW_rTIME()CT</font></td></tr>
            </table>
            <hr>
            %}
```

Figure 44. HTML for the Standard Header

The standard header starts off the HTML document and displays a table with the Grede logo, the company name, the name of the page or application (which is in the title variable), and the current date and time. Variables are replaced with the values that they represent as the program executes. The standard trailer is similarly defined and includes the name of the Net.Data program and ending HTML tags.

The statement after the standard header, `getlist` (**3**), is a function call. Net.Data has a number of very useful and powerful functions. You can also write your own functions. Functions can be written in Net.Data's native language, REXX and Java. Using REXX requires that a REXX interpreter be installed, like Object REXX for Windows. Using Java requires that a Java Development Kit be installed.

Functions also are used when an SQL statement needs to be executed, which is what `getlist` does. Such functions must appear before anything that calls it. That is why the report section is at the bottom of the program. The function that executes the SQL and displays the results (in `report`) must be before it.

As shown, the `getlist` function executes an SQL command and returns the results in a Net.Data table. The table can be displayed using a combination of Net.Data language elements and HTML. The Net.Data language elements allow navigation through the rows and columns of the table, and the HTML is used to format the table elements for display on the Web page.

The six lines of the program after the start of `getlist` is the SQL to be executed. The SQL contains a variable, `$(custnum)` – **4** – which is passed to the program from another Web page.

After the SQL statement is a language element called a `MESSAGE` block (**5**). The message block is used to examine the return code from the SQL and display an appropriate response in the browser. This message block only checks for a return code `+100` (the sign is assumed in the program).

A `+100` is returned by the database when the result set of an SQL statement is empty (no record found). In this case, a message is displayed in red, the function ends, and control is returned to the caller (the report section). All other return codes are lumped into the default block, which displays a generic message indicating that an error occurred and shows the error code. Any Net.Data language statements are valid in a message block.

If the execution of the SQL statement is successful, control drops to the REPORT block (6). It is used to format the result set into HTML for display on the browser. In this case, the report block generates an HTML table tag, a caption tag to display a title for the table, and table row (<tr>) and table header (<th>) tags to display headings for the columns.

Within the report block is a ROW block (7). For each row returned from the SQL statement, the row block is executed once. The HTML in the row block is generated and displayed on the browser. The row block contains table row and table data (<td>) tags to format a row and (in this case) three columns of data – the three columns that were requested in the SQL statement.

The middle column of the displayed table is set up as an HTML link to another Net.Data program. The program name is pattern_detail.mac and the entry point is report. Values from the existing program (V_Plant and V_Pattern_Num) are passed to the new program as HTML parameters.

Notice the V_ that precedes the variable names. In order to be able to handle database column names as variable names, they must be preceded by V_. It is actually not as bad as it sounds. This way, anything preceded by V_ is certain to be a result column name instead of a regular Net.Data variable.

After the result set is processed by the row block, control drops through to the end of the report block, which generates the end table tag and displays a link to another page. After the link is displayed, the function ends. Control returns to the caller, which displays some trailing HTML and the program ends.

When all is said and done, the resulting Web page in Figure 45 on page 101 is displayed. The column in the middle is a hyperlink that takes the user to another Web page with detailed information about that part number.

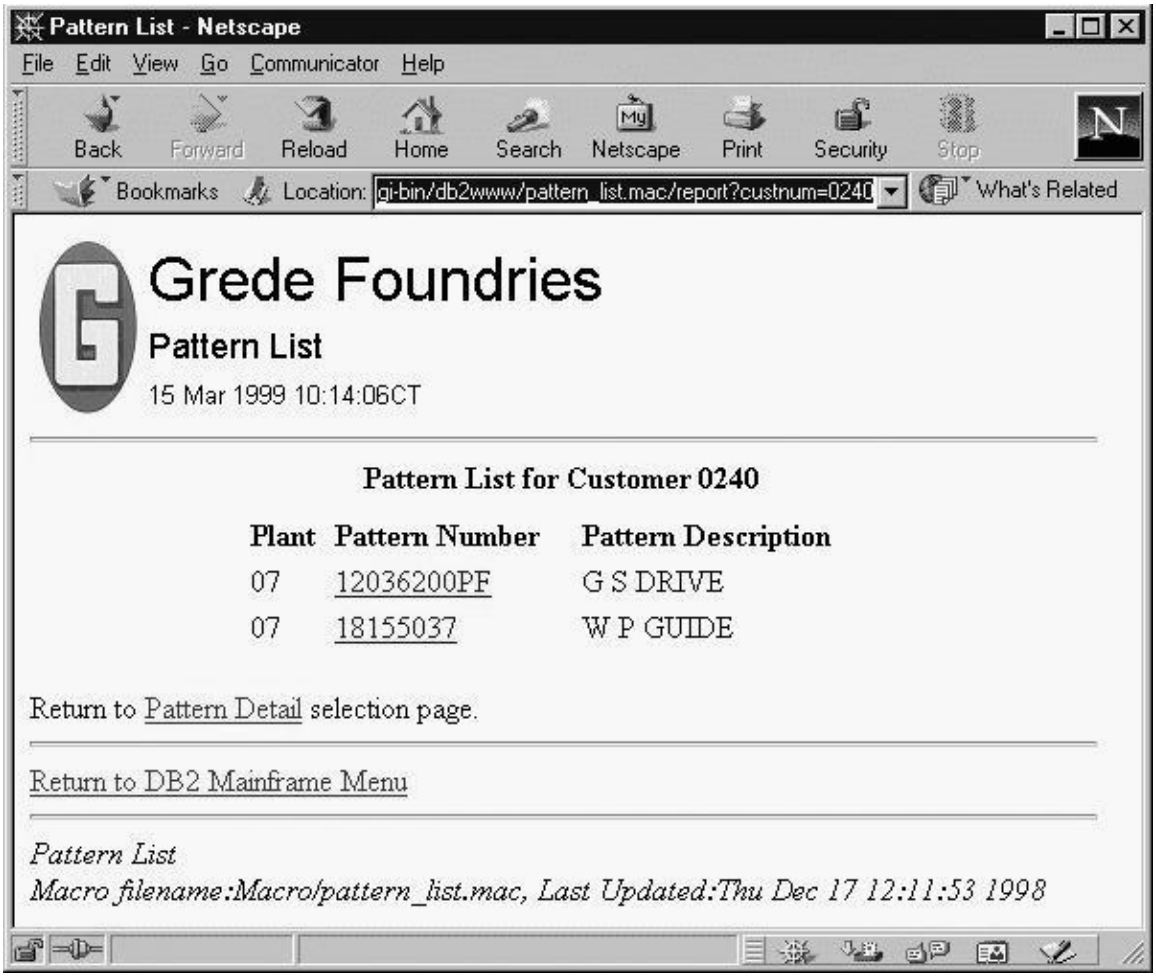


Figure 45. Displayed Web Page

Security is certainly a very large concern here. There are a number of ways that the information can be secured. The database provides table level security. Users must have the correct permissions to access the data. The user IDs are defined to the data base; and if the user ID issuing the query matches a user defined for that table, access is allowed.

Net.Data also can get involved. It would be quite easy to set up a "login" Web page which requests a user ID and a password.

Net.Data has special variables for user ID and password that can then be automatically passed in queries and verified on DB2. Additionally, the Web pages themselves can be secured. When users request access to a page, they are presented with a signon dialog box. This feature is available for all Web servers.

The Web page in Figure 46 on page 102 is a sample of entry fields that are used to qualify some query parameters. The list box is filled in by a database query. Under the expanded list box is a text box that allows user to query the quantity of a part at a specific location.

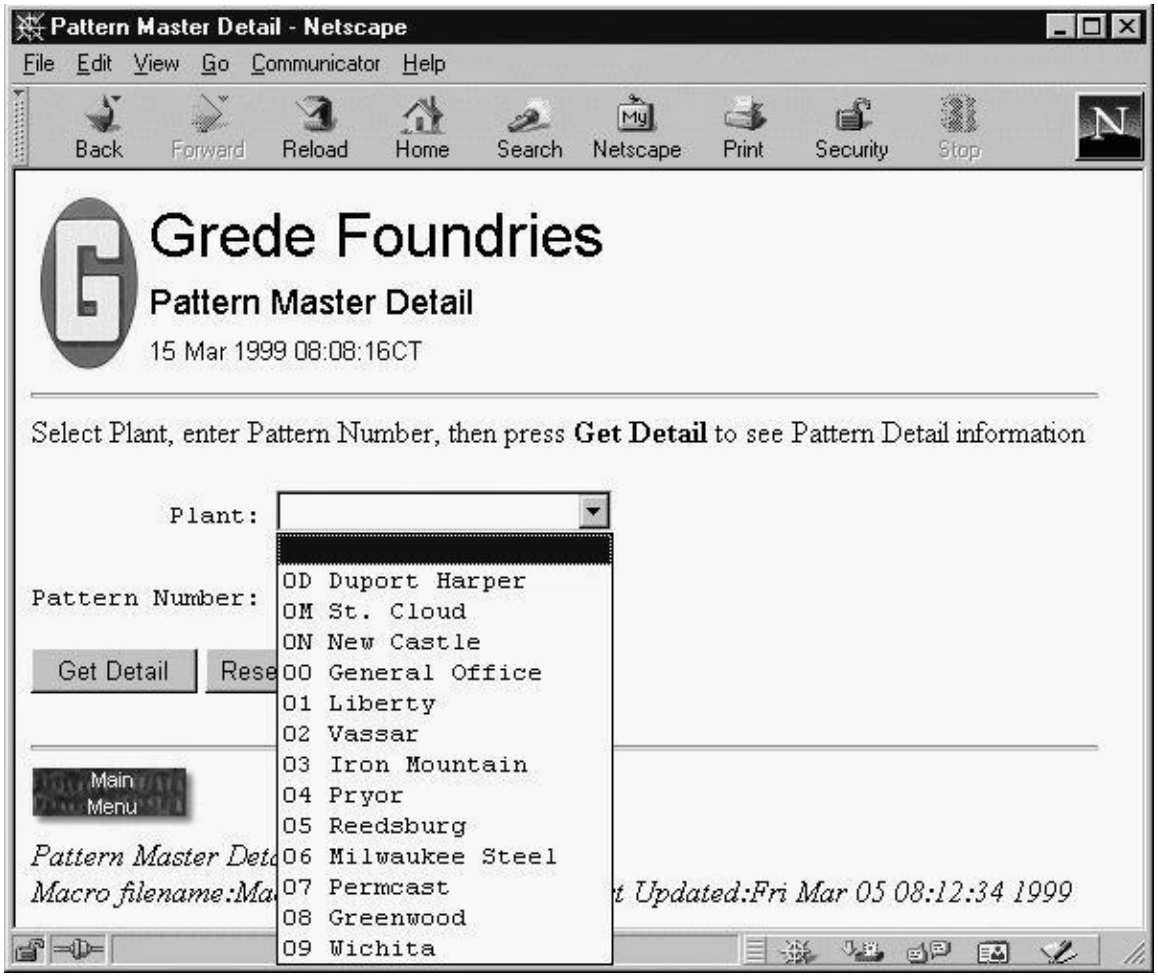


Figure 46. Subsequent Web Page

Conclusion

The combination of these software products provides a very powerful Web serving/application development environment that gives everybody the best of all worlds. Users get their GUI interface; the data is on the mainframe where it belongs; and programmers have an opportunity to use new tools to learn HTML programming and Web page design.

Using ODBC and SQL to Access VSE Data

Author

Randall Evans
Senior Software Engineer
Viaserv Inc.
revans@viaserv.com

When a VSE installation makes the decision to integrate the mainframe with LAN-based and Web-based technologies, the door is opened to a world of possibilities. Integration lets you sustain and enhance your investment in VSE and allows VSE data and applications to participate in one or more of the following activities:

- Web enablement
- PC client access
- e-business
- Multi-tiered OLTP
- Data staging and movement
- Data warehousing

Commercially available middleware plays a key role in the implementation of integrated systems. Without middleware, VSE shops would be faced with a large development effort and would have to master many complicated technologies. Fortunately, there are many alternatives, including 3270 emulation, FTP protocols, screen scrapers, messaging and queuing software, ODBC drivers and gateways, and others.

This article focuses on the use of **Open Database Connectivity** (**ODBC**) and **Structured Query Language** (**SQL**) to access non-relational data sources like VSAM, DL/I, and sequential disk and tape.

Open Database Connectivity

The Open Database Connectivity specification is an API (application programming interface) developed by Microsoft, which defines a standard way for front-end tools and applications to access a relational DBMS.

ODBC access usually begins with an ODBC driver that resides on a client machine. Front-end tools and applications issue ODBC calls to access data from some external data source, and it's the ODBC driver's responsibility to connect to the specified database, execute the SQL request, and return the results and messages to the requesting client application in the expected format. Here's an example of a typical ODBC calling sequence from an application that selects all columns from a table named *Authors*:

1. **SQLExecDirect("select * from authors")** – executes the statement
2. **SQLNumResultCols (...)** – (optional) returns number of columns in result set
3. **SQLDescribeCol(...)** – (optional) describes each column in result set
4. **SQLFetch(...)** – fetches next row of result
5. **SQLGetData(...)** – retrieves data for each column in result set
6. Steps 4 and 5 are repeated until SQLFetch returns "no more data"

ODBC drivers must also support access to the data source's catalog information. ODBC functions like `SQLTables()` and `SQLColumns()` define a standard method for front-end tools and applications to obtain "meta" information about the data source. This capability allows the requesting application to obtain information about table names, column names, column data types, column lengths, primary keys, indexes, and so on. Most ODBC applications normally use one or more of these functions as part of their standard processing.

Front-end Tools and Applications

There are literally hundreds of popular, off-the-shelf tools for query, reporting, and application development that rely on ODBC for access to relational data. With the right middleware, products like Access, Excel, Crystal Reports, Cognos Impromptu, Business Objects, Visual Basic, PowerBuilder, and VisualAge can all be used to access mainframe-based, non-relational data.

The same holds true for Web applications. Whether you are using CGI, Java Script, VB Script, Active Server Pages, or Web development environments like Cold Fusion or Visual Interdev, middleware allows the use of ODBC to build dynamic Web pages using VSE data sources. For applications that use the Java Database Connectivity (JDBC) standard, JDBC to ODBC bridges are available, and some middleware supports the JDBC standard directly.

Connecting to the Mainframe

To access data on a VSE mainframe, the ODBC driver must be capable of connecting to the mainframe directly or to a LAN-based gateway that in turn provides the connection to VSE (Figure 47).

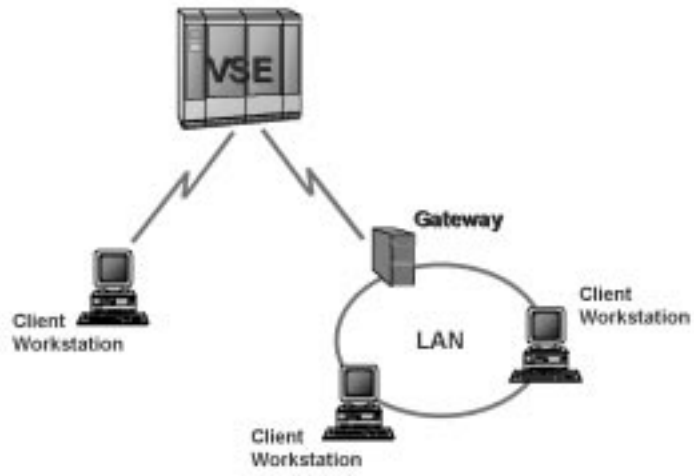


Figure 47. Connecting to the Mainframe

From the ODBC application's perspective, it doesn't really matter whether the associated ODBC driver connects directly or through a gateway. In either case, the results and messages returned to the requesting application are the same. There are, however, some practical considerations that might favor one method over another:

- On the one hand, a gateway-less implementation is simpler in terms of installation and configuration and does not require the use of a separate LAN-based server machine. This method requires that communication software reside on each user machine. For TCP/IP connections, this is usually a trivial requirement; but for LU6.2 connections, communication software like Microsoft's SNA Server or IBM's Communication Manager must be installed and licensed on each client machine.
- On the other hand, in an implementation that uses a gateway, some of the software and much of the processing can be offloaded from the client machine, configuration and administration are centralized, and various kinds of advanced functionality becomes possible.

On the Mainframe

Once the SQL request arrives at the mainframe, other components of the middleware perform the actual processing of the SQL statement.

Some middleware implementations use CICS as the portal into the VSE system. Others use an application running in a separate VSE partition. Using CICS offers certain advantages such as standard CICS security services, a full-functioned online programming API, and the existence of transaction management capabilities.

When accessing a relational DBMS (such as DB2), the DBMS resides in a separate VSE partition. Most of the processing that must be performed by the DBMS is more appropriately done outside the control of CICS. The query may result in a significant amount of file access, or it may require a sort because the query contains an ORDER BY clause.

The same considerations hold true for access to non-relational data. Regardless of the method of entry into VSE, the "relational engine" component of the middleware will reside in a separate VSE partition, and VSE Cross Partition Communication support (XPCC) will be used to communicate between the two partitions.

Basic Functionality

In order for ODBC-compliant tools and applications to access mainframe data, the middleware must provide a minimum set of functionality:

- It should allow either LU6.2 or TCP/IP connections to the mainframe.
- It should support all four of the SQL Data Manipulation Language (DML) statements: SELECT, INSERT, UPDATE, and DELETE.
- At a minimum, it should support the ODBC catalog access functions SQLTables, SQLColumns, SQLStatistics, SQLSpecialColumns, and SQLPrimaryKeys.
- It must translate between EBCDIC and ASCII as appropriate.
- It must be able to convert all S/390 datatypes into datatypes expected by the ODBC application.

If VSE data sources are to be updated by ODBC applications, then the middleware should be capable of performing basic transaction management, automatically issuing COMMITs and ROLLBACKs at suitable points during processing or at the request of the ODBC application.

Mapping Non-Relational Data

In order to use SQL to access a non-relational data source (for example, VSAM), the data must first be mapped into a relational format. The mapping is typically done by the database administrator and must be done once for each "virtual table" that is to be defined.

During the mapping process, the file, its records, and fields in the records are described in terms of relational constructs (tables and columns). The mapping of data into "virtual tables" is usually performed with the use of a windows-based GUI application that lets you specify such things as column name, column length, column datatype, and the column's position in the underlying record. Once the file is mapped, the information is stored – usually in a mainframe-resident catalog – and is used later to satisfy SQL requests against the particular file. Some middleware solutions let you map a file automatically, using existing COBOL copybooks to supply the structure of the file.

The mapping process, to be useful in the real world, must provide ways to map complex data structures like repeating fields, variable length records, and multiple record types. For example, consider a VSAM file with two record types: "A" records containing name information and "B" records containing address information. Each record type is mapped to a separate "virtual table" that can be accessed individually or can be joined with data from other "virtual tables".

Sample VSAM File

Table 2. Sample VSAM File

521-72-2901	A	John	Smith	W	Brown	Brown
521-72-2901	B	1234 Main Street	Denver	CO	80210	
522-80-1245	A	Susan	Johnson	E	Black	Brown
522-80-1245	B	708 University Ave.	Boulder	CO	80303	
523-45-3228	A	Bill	Green	T	Blond	Blue
523-45-3228	B	1882 East 14th Street	Denver	CO	80212	

Table 1 – Name Information

Table 3. NAME_INFORMATION

SSNO	T	FIRST_N	LAST_N	MI	HAIR	EYES
521-72-2901	A	John	Smith	W	Brown	Brown
522-80-1245	A	Susan	Johnson	E	Black	Brown
523-45-3228	A	Bill	Green	T	Blond	Blue

Table 2 – Address Information

Table 4. ADDRESS_INFORMATION

SSNO	T	STREET	CITY	ST	ZIP
521-72-2901	B	1234 Main Street	Denver	CO	80210
522-80-1245	B	708 University Ave.	Boulder	CO	80303
523-45-3328	B	1882 East 14th Street	Denver	CO	80212

Accessing Data

Once the VSE files have been mapped, ODBC applications can access these "virtual tables" just as though they were tables residing in a LAN-based, relational DBMS.

The middleware should support standard relational constructs like multi-table joins, search conditions, and the ordering of result sets. In the following example, two VSAM files are joined on `Social Security Number` to produce a result set of all employees with brown hair, living in Denver.

```
SELECT SSNO, LASTNAME, FIRSTNAME, HOURLY_RATE
  FROM NAME_INFORMATION A, ADDRESS_INFORMATION B
  WHERE A.SSNO = B.SSNO
     AND HAIR = "Brown"
     AND CITY = "Denver"
  ORDER BY LASTNAME, FIRSTNAME
```

When the "relational engine" receives the SQL query, the statement is parsed, access paths to the data are determined, optimization is performed, and the query is executed. The WHERE clause specifies any record selection criteria to be applied. If the underlying files have primary or alternate indexes defined and the structure of the query lends itself to using them, then the middleware should attempt to take advantage of their existence. If the query contains an ORDER BY clause, the result set may need to be sorted before it is returned to the application.

As the rows of data are returned to the requesting ODBC application, the necessary conversions are performed on the data. For character columns, the data is converted from EBCDIC to ASCII. Columns that contain binary data are left unchanged. Columns with packed decimal or floating-point numbers, date, time, or timestamps are converted into their ODBC equivalent.

Modifying Data

ODBC applications that modify data will do so by issuing SQL INSERT, UPDATE, and DELETE statements. As with queries, these statements are processed by the "relational engine" component of the middleware, which parses, optimizes, and executes the SQL statement.

Additional considerations come into play when the middleware is to modify data. First and foremost is the possibility that the files to be updated are already "open" and "in use" by CICS applications. Given that VSE/VSAM Shareoption(4) is usually not a practical option and that closing the files to CICS is probably not a viable alternative, the middleware should have the ability to ship the actual updates to the CICS partition for execution. Updates will be performed under the auspices of the CICS concurrency controls.

Performing the updates in the CICS partition has another important advantage. Some file systems (VSAM, for example) have no native support for COMMIT and ROLLBACK processing. CICS, on the other hand, has robust transaction management support.

Transferring Data

While direct client access to VSE data sources has its benefits, sometimes it may make sense to move VSAM or DL/I data into one or more LAN-based DBMSs and then access the data locally. This may be done for a variety of reasons:

- Reduce the processing load on the mainframe
- Faster response time at the client
- Populate a departmental DBMS
- Take advantage of advanced DMBS functionality
- Build data warehouses or datamarts

Some middleware solutions provide a transfer capability that allows client applications to select VSAM or DL/I data using SQL and then move the data directly into a relational DBMS. Using SQL and middleware to transfer the data has some advantages over traditional FTP methods. First, SQL is easy to use, lets you select specific columns or subsets of records, and allows the use of joins across multiple files. Second, VSE data is likely to contain data (packed decimal for example) that would require special pre- or post-transfer processing. And finally, FTP requires the use of intermediate "flat" files that must be loaded into the target DBMS in a separate step.

Once a shop has elected to offload data to another location, it faces the challenge of keeping the two data stores synchronized. This may entail a periodic refresh performed once a day, once an hour, once a month, or whenever. Or it may entail the capturing of changes that are occurring to the source data and replicating the changes on a real-time or near real-time basis. In either of these cases, the middleware can be used to transport the data to the target location. Sometimes, synchronization must also be performed in the other direction. That is, data will be selected (using SQL) from the LAN-based DBMS and then transferred directly into VSE data stores.

Security Considerations

Typically, when an ODBC application is initiated, it interacts with the end-user to obtain a user ID and password. This information is used by the application to perform a logon to the target DBMS. For applications that access VSE data, the middleware must pass the logon information to the mainframe for validation by CICS or some other mainframe-based security mechanism.

Ideally, the middleware will also provide table and column security in the style of a relational DMBS. Privileges on tables and columns are established with the use of SQL GRANT and REVOKE statements, as shown in the following example:

```
GRANT SELECT ON PAYROLL_MASTER TO USER1
```

ViaSQL for VSE

ViaSQL for VSE, from Viaserv Inc., provides ODBC access to DB2, VSAM, DL/I, sequential disk, and sequential tape and offers most of the functionality outlined in this article. (JDBC support is coming soon.) ViaSQL lets your VSE mainframe participate in all the new and exciting computing technologies.

ViaSQL supports two ways to connect to VSE:

1. A stand-alone ODBC driver allows clients to connect directly to the mainframe, or
2. Clients can connect through a centrally controlled and administered LAN-based gateway.

In addition to the basic ODBC support, a number of advanced features make ViaSQL a full-functioned, flexible middleware solution:

- The built-in TRANSFER feature allows tools and applications to use SQL statements to select records for bi-directional movement of data between VSE and LAN-based data sources. There's direct support of the native APIs for Oracle, Microsoft SQL Server, and Sybase Adaptive Server, and ODBC support for all other relational DBMSs.
- Additional transfer management and scheduling capabilities are available through a Windows-based GUI tool used to create, initiate, manage, and monitor transfer operations. The tool also provides the ability to perform special pre- or post-transfer processing, like sending an email when a transfer completes, for example.
- Remote Stored Procedures (RSPs) allow client-based or web-based applications to execute CICS programs. Remote Stored Procedures can access data sources for which ViaSQL has no direct SQL support, like Ababas, Datacom, IDMS, and others. From the application's perspective, a result set returned by an RSP looks just like a result set returned by an SQL query. RSPs can also be used to perform special manipulation, execute important business logic, or to exercise additional control over OLTP processing.
- Client Services Applications (CSAs) provide CICS applications with SQL access to LAN-based DBMSs like DB2/6000, Microsoft SQL Server, Sybase Adaptive Server, Oracle, and others. You can use CSAs to push data from VSE to the LAN or to pull data from the LAN to the mainframe. Client Services Applications can initiate TRANSFER operations and can even invoke LAN-based programs that are written in C, C++, or Visual Basic.

And there's a myriad of other features like client-idle timeout, a query governor, the ability to configure how datatype conversions are performed, and the ability to do data compression between the mainframe and LAN.

Additional Information

Viaserv is a member of S/390 Partners in Development. For more information about ViaSQL, visit:

http://www.viaserv.com/

Web-Enabling Native VSE Applications Using Web/VSE-Host

Author

Chuck Arney
*Intell*Ware Systems, Inc.
carney@intelliware.com

VSE/ESA's capabilities have extended remarkably over the last several years. This includes improvements in capacity, performance and connectivity. As networks continue to evolve, VSE/ESA is playing several key roles as a data and application server. Since over 85% of the world's data continues to reside on a mainframe, access from today's technologies is imperative. Web-based access to host data (or "Web-to-host") is the way of the future, incorporating the latest Internet/intranet technologies with the stability and robust performance of the mainframe.

This article explores how Web-to-host software works and how it can be implemented to deliver something that users have wanted for years-universal access to data.

What Is Web-to-Host?

A basic definition of a Web-to-host software product is a program that provides access to 3270-based applications from a networked computer attached to the Internet or an intranet. Instead of using old-fashioned 3270 hardware emulator boards or newer TN3270 client software that must be installed on each computer for accessing mainframe applications, Web-to-host uses the emerging standard user interface, the **Web browser**. Since most networked computers (PCs and others) already have this software installed, no additional client software is necessary. That's a big plus when it comes to purchasing, installing and maintaining additional client software on hundreds or thousands of computers. In addition, the Web-to-host software performs its job without application changes. *Absolutely no changes are required to the existing mainframe applications.*

Several different hardware and software platforms are available to provide the Web-to-host connectivity. Some implementations use a "three-tier" client server approach, while others use a native mainframe implementation.

Three-Tier Implementation

When using a three-tier implementation, the Web browser running on the end user's computer connects to a Web server running on another platform (Windows/NT or UNIX, for example). The Web-to-host product also runs on that machine and is executed by the Web server. The product then uses a 3270 emulator or TN3270 client to establish TCP/IP or SNA connectivity to the 3270 mainframe applications.

Thus a minimum of three computers is involved in the connection.

Two-Tier Implementation

The other approach is to remove the middle platform and the Web server that runs there. In this "two-tier" solution, the Web server software is moved to the mainframe, where it is closer to the user application and data.

This approach removes at least one level of complexity and potential point of failure by eliminating the intermediate operating system and associated hardware. By using a "native" implementation in a VSE environment, the management issues associated with multiple platforms is also eliminated.

What Is Web/VSE-Host?

The rest of this article describes the implementation of Web/VSE-Host, a Web-to-host software product from *Intell*Ware Systems. This product is a native VSE/ESA application that executes in a VSE partition. It requires the use of the TCP/IP for VSE product that may be obtained from *Intell*Ware Systems, Connectivity Systems Incorporated, IBM, or other re-sellers around the world. The TCP/IP for VSE HTTP daemon is used as the Web server, providing a complete, native VSE/ESA solution.

The Web/VSE-Host software can be thought of as having two parts:

1. The first part is an interface to a Web browser. The browser user initiates the connection by clicking a link on a Web page or by typing the appropriate information in the browser's URL line.

   This part of the product runs as a Common Gateway Interface (CGI) program. The Web server invokes the CGI program and passes the variable data provided by the Web browser.

   This part of the product is responsible for automatically creating a Web page that contains the data supplied by the 3270 application. The CGI program provides the HTML to the Web server, which sends it to the browser. Refer to Figure 48 on page 112 for an overview of this process.

2. The second part of the product is similar in function to a VSE implementation of a TN3270 client. The information provided by the browser includes the IP address and optional port number of the desired 3270 application server. This is the IP address/port number of a TN3270 server. It is not limited to the VSE Telnet server. It may be any VSE, VM, or OS/390 server, allowing the VSE system to be a gateway to other networked computers.

   This part of the product uses the TCP/IP for VSE assembler Sockets API to establish and manage connections with TN3270 servers. The 3270 data streams are transmitted between Web/VSE-Host and the TN3270 server. The TN3270 server provides access to the 3270 application.

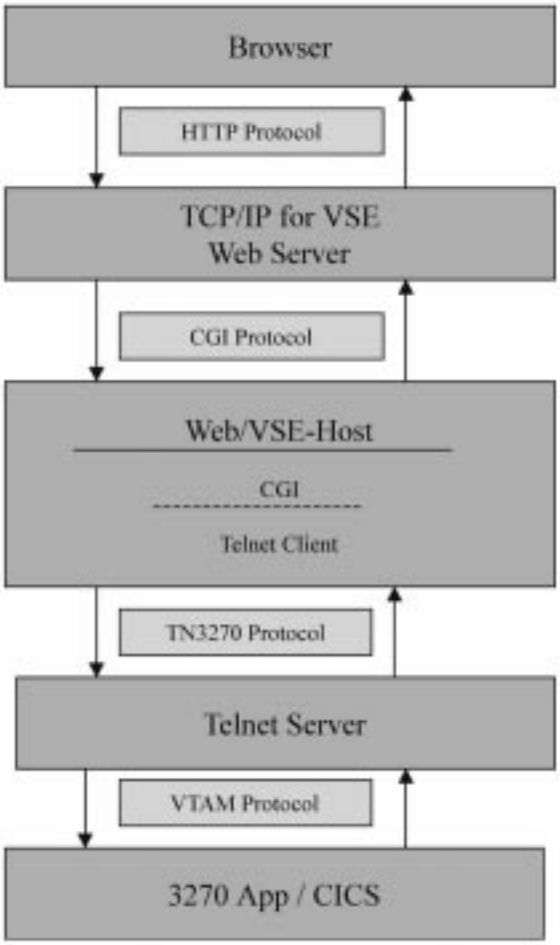


Figure 48. Overview to How Web/VSE-Host Works

The two parts of the product – along with the browser, the Web server, the TN3270 server and the 3270 application – comprise the communications path between the application and the end user.

Data streams received from the application are dynamically converted into Web pages. The pages are created using an HTML form. Each protected 3270 field becomes displayable text on the page and each unprotected field becomes an input text box. The text box may contain data or it may be empty. In addition, the page contains a set of buttons used to create a virtual keyboard. These buttons are used in place of the 3270 specific keys such as PF and PA keys and the Clear and Enter keys.

After a user types any desired data in the input text boxes and clicks the appropriate key, the data is transferred back to the CGI program. Web/VSE-Host uses the browser data to construct an inbound 3270 data stream, and the data stream is delivered to the application.

The status of modified data is maintained and accurately reported to the 3270 application when it issues a read-modified data request. Also, the 3270 cursor and the browser insert cursor position are automatically synchronized so the browser user does not have to manually position the cursor in normal circumstances, and the 3270 application can determine where the user positioned the cursor.

Web/VSE-Host also provides support for extended data streams for full color and special effects that can be translated to the constructed Web page. Figure 49 on page 113 and Figure 50 on page 113 illustrate how the VSE/ESA Function Selection menu might look in a TN3270 client and in a Web browser.

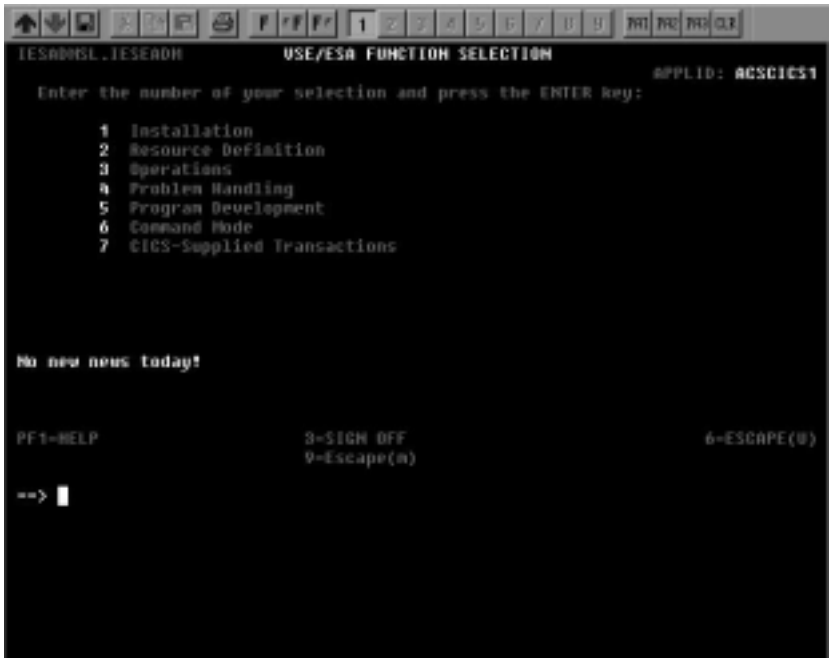


Figure 49. Function Selection Menu – TN3270 Client

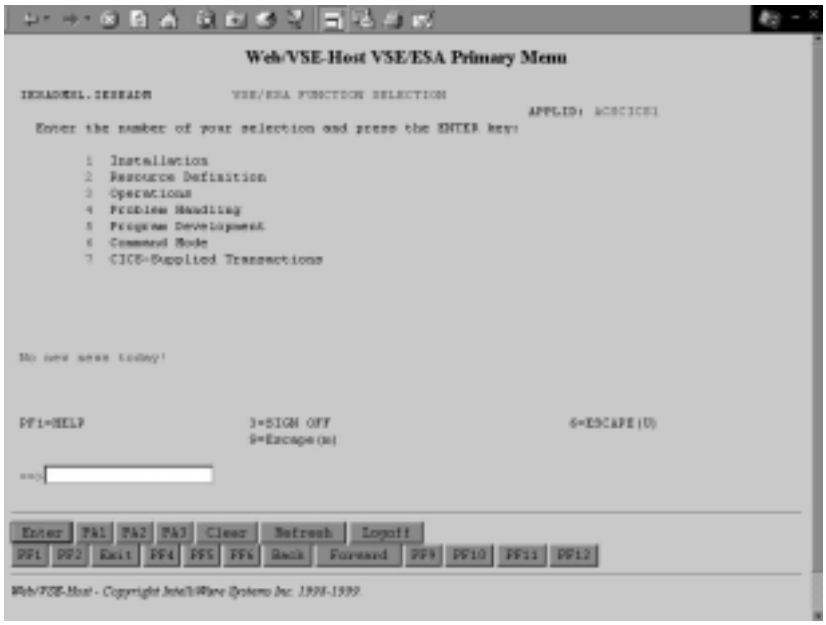


Figure 50. Function Selection Menu – Web Browser

Installation Steps

The implementation of Web/VSE-Host in your data center may consist of some or all of the following steps. If you are already using TCP/IP for VSE, its TN3270 server and Web server, you have already performed some of the installation tasks.

1. Install and customize TCP/IP for VSE.
2. Define, configure and test the Web server.
3. Define, configure and test the Telnet servers and VTAM definitions.
4. Install and configure Web/VSE-Host.
5. Create a Web page containing links to available 3270 applications.

The installation of TCP/IP for VSE is the first requirement. After it is up and running and the Web server is operational, you must configure Telnet daemons. Unless you have specific requirements, a pool of shared Telnet sessions may be used by Web/VSE-Host. If you already have Telnet sessions defined, the existing definitions can be used, including standard or extended data stream terminal connections. If you are using a Telnet server that uses VTAM, be sure to define and activate the VTAM Logical Units.

After the Web/VSE-Host software is installed and the server partition started, it is ready for use. Although it is possible to enter a call to the CGI program from the browser's URL line, its not something you want to do often.

Your next step is to create a Web page. This page will be a menu from which users can choose their 3270 application. If its your first Web page, don't worry about it. Just use the distributed sample page, modify it, and have fun testing and altering the HTML until it looks the way you want. This can be a simple Web page if desired, containing application links defined using the HTML Anchor tag; or it may be a more complex page using an HTML form. The distributed sample page uses a form to provide a set of radio buttons. The user selects the desired application by selecting the proper button. This page will be the starting point for your users. They should always be able to open this page and click a link to start a new 3270 application session.

After your initial use of the product and simply translating the 3270 screens to simple Web pages, you may wish to exploit some of the additional capabilities. Web/VSE-Host's HTML template files may be used to create customized Web pages. The background and text colors may be altered, or a background image used. The color mapping facility may be used to redefine the original 3270 colors to display the color scheme of your choice. On 3270 screens that don't use extended color, you also can choose the color used for highlighted text.

Accessing Multiple Applications

Web-to-host software may provide benefits you have not yet considered. It is more than a simple terminal replacement. Using the Web browser as a user interface has benefits other than the common look and navigation.

Most of today's browser software supports multiple windows that may be open and operating independently. Many data processing installations use session manager software products to provide access to multiple applications from a single physical terminal.

The session management software is not required with Web-enabled applications. Simply open a new browser window for each new application. The application windows may be opened, used, and closed completely independent of each other. Of course, if you must help users keep in touch with the past, the session manager can be used along with Web/VSE-Host to provide the same application "feel" they expect.

Connecting Users to a Specific Application

What do we do about those users that must be automatically connected to a specific application? In the past, VTAM allowed us to "LOGAPPL" our terminals so the user saw only one application. When we switch to TN3270, the TCP/IP for VSE Telnet server even provides this capability when an IP address is specified on the daemon definition.

The same technique can be accomplished with Web/VSE-Host. Web/VSE-Host allows you to specify which Telnet port number a networked computer will use to connect to the Telnet server. Then all you have to do is define a Telnet daemon that listens on the desired port. The daemon definition specifies the 3270 application that is used for the connection. Using this facility, when the user requests a connection with the Telnet server, he/she is automatically connected with the required application.

Web Enabled Interfaces – the Final Frontier

Web-based interfaces are rapidly becoming the de facto standard for users on all platforms. One needs to only look to Windows 98 and the change of the standard Windows interface to a more Web-based look and feel as proof.

With the advent of non-traditional devices including WebTV, Web kiosks at airports and shopping areas, and cell phones with Web capabilities, the Web as an interface is likely to become the most widely spread interface throughout the world. For continued exploitation of today's technologies VSE/ESA must exploit this Web-based explosion through Web-to-host solutions like Web/VSE-Host to stay on the cutting edge.

Additional Information

For more information about WEB/VSE-Host, visit:

http://www.intelliware.com/

More Articles and News

How Can You Find VM or VSE Solutions for Your Business?

Author

James Kingman
Solution Developer Marketing
jkingman@us.ibm.com

How can you find VM or VSE solutions for your business? Search the *Global Software Solutions Guide* – an online catalog containing hundreds of VM and VSE solutions from our partners.

One Source, Many Solutions

The *Global Software Solutions Guide* is an extensive online catalog of applications, tools and services developed using IBM technologies. You can search this Web site any time, day or night, to find solutions to your business challenges. This comprehensive source of software solutions contains over 36,000 entries, many of which are sure to meet your needs.

Listings in the *Global Software Solutions Guide* include product descriptions, company information and direct links to solution providers' Web sites and e-mail addresses (where available).

The *Global Software Solutions Guide* is your one-stop resource for business solutions. Multi-criteria, free-form text searches empower you to evaluate and compare solutions:

Search on a Variety of Criteria

- industry (manufacturing, banking, etc.)
- technology (Year 2000 ready, Java, etc.)
- solution types (applications, tools, services)
- language
- country of availability

Multiple Platforms and Technologies

- VM and VSE
- AIX
- AS/400®
- CICS
- DB2
- Java
- MQSeries
- RS/6000®
- S/390
- Netfinity®
- Windows NT
- And more

Native Languages Available

- English
- French
- German
- Spanish
- Brazilian Portuguese
- Japanese

It's as Simple as 1-2-3

1. Log on to the *Global Software Solutions Guide* at:
   http://www.software.ibm.com/solutions/isv/
2. Select a language and search for solutions by using state-of-the-art queries based on your own criteria.
3. Connect directly to thousands of solutions through links to the providers' Web sites and e-mail addresses. Go from the catalog listing to the product home page with one simple click!

Questions?

If you need assistance or have questions about the *Global Software Solutions Guide*, contact IBM.

Phone:
1-800-627-8363 U.S. & Canada
+1-770-835-9902 Worldwide
(available 8AM - 8PM EST, New York)

e-mail:
ibmsdp@us.ibm.com
USA and Canada

sdpemea@uk.ibm.com
Europe, Middle East and Africa

lasdp@vnet.ibm.com
Latin America

apsdp@au1.ibm.com
Asia Pacific

Nominate Your Solutions and Win with IBM Solution Showcase!

The **IBM Solution Showcase** recognizes outstanding commercial members of the IBM Solution Developer Program who have developed innovative e-business solutions using IBM technologies. Winners will receive up to $30,000 in marketing assistance and technical support. For complete details about the IBM Solution Showcase or to nominate your solutions online, visit:

http://www.developer.ibm.com/welcome/showcase.html

VM/ESA and VSE/ESA Technical Conference

Author

Sharon Kent
Learning Services, IBM Global Services
smkent@us.ibm.com

VM and VSE customers who couldn't attend the 1999 VM/ESA and VSE/ESA Technical Conference in Orlando, Florida, won't want to miss the VM/ESA and VSE/ESA Technical Conference in Mainz, Germany. In Mainz, see how VM/ESA and VSE/ESA can take you into the year 2000 and beyond using products, tools, and services from IBM and others! Choose from more than 60 elective sessions that give you valuable hints, tips, and recommendations to help you make the most of your VM/ESA and VSE/ESA system. Our rich agenda focuses on these key areas – *network computing*, *security*, *performance*, *business intelligence*, *e-business*, and *application development*.

Our Product Expo gives you a first-hand look at the most current VM and VSE hardware technology from IBM and other vendors. We also offer two hands-on workshops:

1. TCP/IP for VSE/ESA™ – learn how to set up, customize, and use TCP/IP for VSE/ESA.
2. NetRexx™ on VM/ESA – learn how to develop Java applications with NetRexx on VM/ESA.

Plus, there's an optional, half-day, no-charge DFSORT/VSE™ class on June 17.

Mark your calendars for the VM/ESA and VSE/ESA Technical Conference, June 14-16 in historic, charming Mainz, Germany. For detailed conference information and to enroll, visit our Web site:

http://www.ibm.com/services/learning/conf/vmvse/

Figure 51. Scientia Est Potestas!

IBM VM/ESA and VSE/ESA Customer Conference Calls

Author

Julie Liesenfelt
VM/ESA & VSE/ESA S/390 Systems Center
juliev@us.ibm.com

The **IBM Enterprise Connection Conference Call Series** makes it easy to keep up with the latest developments on VSE and VM and other enterprise-level technology. You only need a phone to participate.

Each hour-long call features an expert on the topic at hand – a developer, specialist, consultant, or customer. The call begins with a presentation which is usually followed by a question and answer session that you can participate in.

While all these calls are offered at no-charge, you must register to attend. Upon enrollment, you will receive a dial-in number for the call and a choice of easy options for receiving your presentation handouts for use during the call.

Upcoming Calls in 1999

This is the current list of upcoming VSE and VM Calls:

05/12/99	VSE	Plugging VSE/ESA into the Net
06/23/99	VM	VM Application Development Hints and Tips
08/26/99	VSE	VSE/ESA V2.4 Security
09/15/99	VSE	Migrating to CICS Transaction Server for VSE/ESA
09/22/99	VM	VM/ESA Release Update and Customer Experiences
10/20/99	VSE	A Progress Report on VSE/ESA
10/27/99	VM	Data Management and Business Intelligence with VM/ESA
11/16/99	VM	Getting More From e-business with VM/ESA
11/17/99	VSE	Overview of the CICS Web Interface and 3270 Bridge

Please note that this schedule and topics are subject to change, so please check this Web site for the very latest information on this call series:

http://www.ibm.com/s390/events/

You can enroll for calls from the Web site, view call presentations, or view a list of previously held calls that can be requested on audio tape. And new in 1999, you can request audio replay.

Audio Replay

If you miss a call, you can listen to the audio replay over the Internet. Just go to the Web site shown above, and look under "previously held conference calls" to see the IBM Enterprise Connection Presentation Library.

The use of this new service requires that you check-in before entering the library. Thus you will have to register beforehand. We will post previously held calls for up to six months, and most replays will be posted within 72 hours of the call date.

In the library, you can see the presentation by clicking on the PDF icon. To listen, click on the Audio Replay icon. (If you don't have RealAudio installed, you can click to download it.)

Previously Held 1999 VSE and VM Conference Calls

01/20/99	VSE	VSE/ESA Performance Top Hits
02/17/99	VSE	T-317 Days and Counting to the Year 2000
02/24/99	VM	IT Savings via Server Consolidation and VM/ESA
03/17/99	VSE	Basics of Internet Security
03/24/99	VM	VM Platform Update- Guest Support and Future Directions
04/28/99	VM	Expanding e-business Options

Thanks for listening!

Vendors of Related Products

Author

Denise A. Battaglini
S/390 VM and VSE Techincal
Marketing Support
dbattagl@us.ibm.com

The table below is an update to the information that appeared in the last issue of the *Newsletter*. It's from Denise Battaglini, who is a member of Len Diegel's VM and VSE Technical Marketing Support Team. Please contact **both** Denise and the *Newsletter's* editor (jhermann@de.ibm.com) if you have additions, changes, or corrections. For more information or for products not contained in this table, see the *S/390 Application Solutions Directory* on the Web:

http://www.ibm.com/software/solutions/isv/

Vendor/Company Web Page	Product	Industry Category	Platform VM/VSE
Allen System Group www.allensysgroup.com	ASG-LDS	Cross Industry	VSE, MVS
Allen Systems Group www.allensysgroup.com	ASG-XPATH	Cross Industry	VM, VSE, MVS
American Software www.amsoftware.com	The Enterprise (ERP Application)	Cross Industry	VSE
Aonix www.aonix.com	RP/Web	Cross Industry	VM
Aonix www.aonix.com	RP/Server	Cross Industry	VM
Aonix www.aonix.com	NOMAD	Cross Industry	VM
ASCENT Solutions inc. (ASi) www.asizip.com	PKZIP VSE	Cross Industry	VSE
ASCENT Solutions inc. (ASi) www.asizip.com	PKZIP VM	Cross Industry	VM
Beyond Software Inc. www.beyond-software.com	EnterpriseWeb/VM	Cross Industry	VM Support: World Wide
Beyond Software Inc. www.beyond-software.com	EnterpriseWeb/MVS	Cross Industry	MVS
Beyond Software Inc. www.beyond-software.com	Enterprise VIEW	Cross Industry	VM
Beyond Software Inc. www.beyond-software.com	EnterpriseWeb Calendar	Cross Industry	VM
Beyond Software Inc. www.beyond-software.com	EnterpriseWeb Secure	Cross Industry	VM
BMC Software www.bmc.com/	3270SuperOptimizationr/VM	Cross Industry	VM
Circle Computer Group NA	ROLF/BRF	Cross Industry	VSE
Circle Computer NA	REPRO	Cross Industry	VSE
Circle Computer Group NA	Xpress	Cross Industry	VSE
Circle Computer Group NA	Audit 2000	Cross Industry	VSE
Circle Computer Group NA	COMvert	Cross Industry	VSE

Vendor/Company Web Page	Product	Industry Category	Platform VM/VSE
cfSOFTWARE, Inc. www.cfsoftware.com	XAPI (Terminal & communications control interface)	Cross Industry	VM, VSE
cfSOFTWARE, Inc. www.cfsoftware.com	DIALOG (Advanced 3270 scripting facility)	Cross Industry	VM, VSE
cfSOFTWARE, Inc. www.cfsoftware.com	APPX (Peer-to-peer communications channel)	Cross Industry	VM, VSE
cfSOFTWARE, Inc. www.cfsoftware.com	pcMAINFRAME (Strategic tool to join PC, LAN, and mainframe worlds)	Cross Industry	VM, VSE
Computec www.computec.com.ar	MANTEC-WIN Computed Maintenance Management System	Cross Industry	VM, VSE Latin America
Computec www.computec.com.ar	Catalog WIN 2.0. Materials Cataloging	Cross Industry	VM, VSE Latin America
Computec www.computec.com.ar	SUMTEC-WIN Purchase, Materials, and contract management.	Cross Industry	VM, VSE Latin America
Computec www.computec.com.ar	ISOMANAGER Administration/Documentation for ISO9000 activities	Cross Industry	VM, VSE Latin America
Control Software Inc. www.csoft.com	Maintenance Control & Management System (MCMS)	Transportation, Public Sector, Construction Utilities Cross Industry	VSE Support in England, Saudi Arabia
Council EDV- und Managementberatung Gesellschaft mbH www.council-systems.com	Virtual Machine System Manager	Cross Industry	VM, VSE
Cross Access Corporation www.crossaccess.com	Data Mapper	Cross Industry	VSE
Cross Access Corporation www.crossaccess.com	Cross Access	Cross Industry	VSE
Janake K. Dabare NA	Job Scheduler	Cross Industry	VSE
Data21 www.data21.com	IP Server for CICS (E-business appl)	Cross Industry	VSE
Decision Technology Inc. www.dtiprinceton.com	Decision Analyzer	Cross Industry	VSE
DETEC(Decision Technology Software www.det.de	Lasersoft/VSE (Printing facility, Fax)	Cross Industry	VSE
Firesign Computer www.firesign.com/9plane.htm	Outbound	Cross Industry	VM, VSE

Vendor/Company Web Page	Product	Industry Category	Platform VM/VSE
FTB Systems Inc. www.ftbsystems.com	3Comm Autopilot	Cross Industry	VM
Global Financial NA	ManTec	Finance	VSE
GT Software, Inc. www.gtsoftware.com	Assist/GT	Cross Industry	VM, VSE
GT Software, Inc. www.gtsoftware.com	BMS/GT	Cross Industry	VM, VSE
GT Software, Inc. www.gtsoftware.com	READ/GT	Cross Industry	VM, VSE
GT Software, Inc. www.gtsoftware.com	Interactive Books and Screen Partner Mainframe (SPMF)	Cross Industry	VM, VSE
GT Software, Inc. www.gtsoftware.com	Screen Partner Mainframe	Cross Industry	VSE
IntelliWare Systems, Inc. www.intelliware.com	Data Center Management	Cross Industry	VSE
IntelliWare Systems, Inc. www.intelliware.com	OS Programmer Tools	Cross Industry	VSE
IntelliWare Systems Inc. www.intelliware.com	WIN/VSE (Application development)	Cross Industry	VSE
International Software Company NA	AOII (Console Automation)	Cross Industry	VM, VSE
Kuhn & Weyh Software GmbH www.kwsoft.freinet.de	MTEXT (text processing for German documents)	Cross Industry	VM, VSE Germany
Kuhn & Weyh Software GmbH www.kwsoft.freinet.de	M/MAIL (Electronic mail and workgroup component for mainframe)	Cross Industry	VM, VSE Germany
Kuhn & Weyh Software GmbH www.kwsoft.freinet.de	M/AFP (PostScript printing on the mainframe)	Cross Industry	VM, VSE Germany
Kuhn & Weyh Software GmbH www.kwsoft.freinet.de	M/Postscript (PostScript printing on the mainframe)	Cross Industry	VM, VSE Germany
Kuhn & Weyh Software GmbH www.kwsoft.freinet.de	M/ARCHIVE (Document management system for corporate solutions)	Cross Industry	VM, VSE Germany
Landmark Systems www.landmark.com/home.htm	The Monitor for CICS	Cross Industry	VSE
Landmark Systems www.landmark.com/home.htm	The Monitor for VSE	Cross Industry	VSE
Landmark Systems www.landmark.com/home.htm	VM Contention Monitor	Cross Industry	VM, VSE
Landmark Systems www.landmark.com/home.htm	NaviGraph	Cross Industry	VSE

Vendor/Company Web Page	Product	Industry Category	Platform VM/VSE
Life Insurance Data Processing www.lidp.com	The Administrator	Finance, Insurance	VSE
Life Insurance Data Processing www.macro4.com	LIPD's Agency Management System	Finance, Insurance	VSE
Macro4 Inc. www.macro4.com	CICSPRINT and VTAMPRINT	Cross Industry	VM, VSE
Macro4 Inc. www.macro4.com	REFLECT	Cross Industry	VM, VSE
Macro4 Inc. www.macro4.com	CMAP	Cross Industry	VM
Macro4 Inc. www.macro4.com	COMMAND VSE	Cross Industry	VSE
Macro4 Inc. www.macro4.com	DATABACK	Cross Industry	VSE
Macro4 Inc. www.macro4.com	DESKPAD	Cross Industry	VM, VSE
Macro4 Inc. www.macro4.com	DUMPMASTER	Cross Industry	VM
Macro4 Inc. www.macro4.com	EnterWEB	Cross Industry	VM, VSE
Macro4 Inc. www.macro4.com	FOREMAN	Cross Industry	VSE
Macro4 Inc. www.macro4.com	LIBRA	Cross Industry	VSE
Macro4 Inc. www.macro4.com	LOCKMASTER	Cross Industry	VM
Macro4 Inc. www.macro4.com	LOGOUT	Cross Industry	VSE
Macro4 Inc. www.macro4.com	REFLECT	Cross Industry	VSE
Macro4 Inc. www.macro4.com	SYSTEM ACCOUNTING	Cross Industry	VSE
Macro4 Inc. www.macro4.	SYNCHRO	Cross Industry	VSE
Macro4 Inc. www.macro4.com	TUBES	Cross Industry	VM, VSE
Macro4 Inc. www.macro4.com	VPAC	Cross Industry	VM
Macro4 Inc. www.macro4.com	VSAMLITE	Cross Industry	VSE
Macro4 Inc. www.macro4.com	VSAMTUNE	Cross Industry	VSE

Vendor/Company Web Page	Product	Industry Category	Platform VM/VSE
Manager Software Products Limited www.manager-software -products.co.uk	METHODMANAGER	Cross Industry	VM, VSE
Manager Software Products Limited www.mspusa.com	CONTROLLMANAGER	Cross Industry	VM, VSE
Manager Software Products Limited www.mspusa.com	DATAMANAGER	Cross Industry	VM, VSE
Manager Software Products Limited www.mspusa.com	DICTIONARYMANAGER	Cross Industry	VM, VSE
Manager Software Products Limited www.mspusa.com	DESIGNMANAGER	Cross Industry	VM, VSE
Medtrak NA	Emergency Department	Health	VSE
Medtrak NA	Occupational Management	Health	VSE
NBS Systems, Inc. NA	TOSS Mainframe Office Automation for VM/VSE	Finance, Insurance, Banking, Medical, Manufacturing, Transport	VM, VSE
NBS Systems, Inc. NA	Total-FAX	Cross Industry	VM/VSE
NBS Systems, Inc. NA	Golden Mailbridge	Cross Industry	VM/VSE
OptiSystems NA	Energizer for CICS	Cross Industry	VSE
OptiSystems NA	Energizer for R/2-VSE	Cross Industry	VSE
Pacific Systems Group www.optisystems.com	Spectrum Writer VSE 4GL Report Writer	Cross Industry	VSE
Phoenix Software International NA	Auto/FT (Unattended file transfer across multiple platforms)	Cross Industry	VSE
Phoenix Software International NA	CONDOR (Online package of program development, library management and system maintenance tools)	Cross Industry	VSE
Polaris Communications www.polariscomm.com	WindowsNT to Mainframe Connectivity	Cross Industry	VM, VSE
Praim www.praim.com www.docsuite.com	DocSuite	Cross Industry	VM, VSE

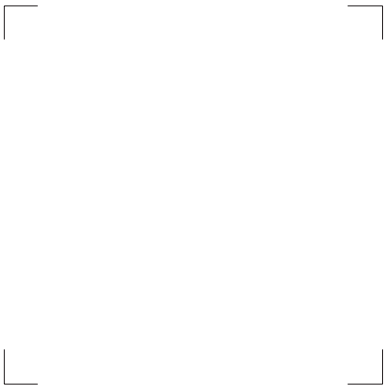
Vendor/Company Web Page	Product	Industry Category	Platform VM/VSE
Safe Software Inc. www.vm.safesoftware.com	SafeSFS	Cross Industry	VM
Sapiens International Corporation, N.V. www.sapiens.com	SAPIENS ObjectPool	Cross Industry Client/Server development toolset	VM, VSE Worldwide support
Sapiens International Corporation, N.V. www.sapiens.com	SAPIENS Object Modeler (Client-based graphical modeling environment for visual application development)	Cross Industry	VM, VSE Worldwide support
Sapiens International Corporation, N.V. www.sapiens.com	SAPIENS WorkStation (Generates GUI clients for ObjectPool applications)	Cross Industry	VM, VSE Worldwide support
Sapiens International Corporation, N.V. www.sapiens.com	SAPIENS Year 2000 Toolkit	Cross Industry	VM, VSE Worldwide support
SEA Software Engineering of America www.seasoft.com	TRMS "Total Report Management Solution"	Cross Industry	VSE
SEA Software Engineering of America www.seasoft.com	CSAR "Computer Scheduling And Reporting system"	Cross Industry	VM
SEA Software Engineering of America www.seasoft.com	KEYS "Complete information database for data center management of MVS and VSE environments"	Cross Industry	VSE
SEA Software Engineering of America www.seasoft.com	PRO-2 "Application generator enabling development of production ready CICS COBOL code"	Cross Industry	VSE
SEA Software Engineering of America www.seasoft.com	CODEC "Non-destructive compressing/ decompression and encryption/ deciphering of files"	Cross Industry	VM, VSE
SEA Software Engineering of America www.seasoft.com	TAPE2000 "Automated Tape Library Management System"	Cross Industry	VSE
Software Engineering GMBH NA	Pilot for Sessions-VM	Cross Industry	VSE
Software Engineering GMBH NA	Online Transmission Time Optimizers (OTTO) for CICS-VSE	Cross Industry	VSE
Software Engineering GMBH NA	Compress for SAP-VSE	Cross Industry	VSE
Software Engineering GMBH NA	Robot for Transactions	Cross Industry	VSE

Vendor/Company Web Page	Product	Industry Category	Platform VM/VSE
SOPRA NA	Inter.Pel v.6.3	Cross Industry	VM, VSE
SOPRA NA	Peli.VM v.2.10	Cross Industry	VM
SOPRA NA	Peli.VSE v.5.20	Cross Industry	VSE
S + S SoftwarePartner GmbH NA	ISF - Financial Accounting (Version G1)	Cross Industry	VSE Czech Rep., Germany Netherlands, Spain
S + S SoftwarePartner GmbH NA	ISA - Fixed Asset Administration (Version G1)	Cross Industry	VSE Czech Rep., Germany Netherlands, Spain
S + S SoftwarePartner GmbH NA	GPR - Profit Planning (Version G1)	Cross Industry	VSE Czech Rep., Germany Netherlands, Spain
Sterling Commerce www.sterlingcommerce.com	CONNECT:Direct	Cross Industry	VM, VSE
Sterling Software www.vm.sterling.com	VM:Account	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Archiver	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Backup	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Batch	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:DBA	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:DB/Admin	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:DB/Editor	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:DB/Monitor	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:DB/Reorganizer	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:DB/Reporter	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:DB/Restore	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:DB/REXX	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Director	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Manager	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Migrate	Cross Industry	VM

Vendor/Company Web Page	Product	Industry Category	Platform VM/VSE
Sterling Software www.vm.sterling.com	VM:Monitor	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Notekeeper	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Operator	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Prorexx	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Schedule	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Secure	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Sort	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Spool	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Tape	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Vantage	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Webgateway	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Webgateway for VSE	Cross Industry	VM
Sterling Software www.vm.sterling.com	VM:Webgateway OfficeVision	Cross Industry	VM
StoneHouse & Co. NA	MONIES Extended System	Finance	VM, VSE
SupplyTech NA	STX for the Mainframe	Cross Industry	VSE Worldwide support
SWAN www.DATACOOP.com	$VISION_{64}$ (formerly DATACOOP)	Cross Industry	VSE, VM USA, France UK, Belgium, Italy
Systema NA	CUBIS "Business engineering optimization of key customer oriented processes"	Cross Industry	VSE, VM
Systema NA	IRD "Customer information and billing system"	Cross Industry	VSE, VM
Systema NA	KARAT Energy "Integrated total business solution for local and regional utilities"	Cross Industry	VSE, VM

Vendor/Company Web Page	Product	Industry Category	Platform VM/VSE
Systematic Software Solutions NA	ISRS "Trickle-Poll Software"	Retail	VSE
Systematic Software Solutions NA	STore Transmission System/1 (STS/1)	Retail	VSE
Systematic Software Solutions NA	Host Positive Credit System	Retail	VSE
Target Four Limited www.target4.com	PKZIP	Cross Industry	VSE
Teubner & Assoc. www.teubner.com	A-Net	Education, Finance, Health, Manufac-turing Retail, Transportation, Wholesale	VM, VSE
The Fredrick Group Inc. www.fredrickgroup.com	TFG 2000, TFG 3000 (ERP Apps.)	Cross Industry	VM, VSE
TRAX Softworks, Inc. NA	Electronic Spreadsheet System (ESS)	Cross Industry	VM, VSE Worldwide support Languages: French, German, English
TRAX Softworks, Inc. NA	EdWord	Cross Industry	VM, VSE
Velocity Software www.velocity-software.com	ESAMAP	Cross Industry	VM
Velocity Software www.velocity-software.com	ESAMON	Cross Industry	VM
Velocity Software www.velocity-software.com	ESATCP	Cross Industry	VM
VIASERV, Inc. www.viaserv.com	VIASQL for VSE - SQL/DS	Cross Industry	VSE Worldwide support
VIASERV, Inc. www.viaserv.com	VIASQL for VSE VSAM	Cross Industry	VSE Worldwide support
VTA Software Inc. www.vtasoft.com	Macro Level Interpreter	Cross Industry	VSE, OS/390
VTA Software Inc. www.vtasoft.com	Macro Level Detector	Cross Industry	VSE, OS/390
VTA Software Inc. www.vtasoft.com	Y2K Simulator	Cross Industry	VSE, OS/390
VTA Software Inc. www.vtasoft.com	ALIAS for CICS	Cross Industry	VSE, OS/390
WINCAP, Inc. www.wincapsoftware.com	WINCAP	Education, Health, Manufacturing, Retail, Transportation Travel, Wholesale Distribution	VSE

Printed in Germany on chlorine-free bleached paper



G225-4508-18