Volume 4, Number 1
December 1998
The IBM VisualAge Generator Newsletter

VisualAge® Generator

A Powerful New Vision of Programming™

Contents

A New Product Manager and the Year in Review

by A. Hayden Lindsey, VisualAge Generator Product Manager

Once again, a new leader is at the helm of VisualAge Generator! Let me introduce myself, at least in my new role as the Product Manager for VisualAge Generator. I am Hayden Lindsey, and I took the reins of the VisualAge Generator area from Peter Spung in June. I would like to thank Peter for his stewardship in the first six months of the year, and wish him well as he returns to being the VisualAge Smalltalk Product Manager.

If my name sounds familiar, it might be due to the fact that my entire IBM career (13 years and counting!) has been in the CSP, VisualGen and VisualAge Generator product areas. I started out building the PC DOS generator and interpreter, helped further our move to the workstation and made a foray into object technology with the Interactive Test Facility. In addition, I have been one of the area architects in the last few years. In the past 13 years, I have had the opportunity to meet many of you and hope to expand these interchanges to ensure that we are taking VisualAge Generator in the directions that are consistent with your needs and your visions.

In this edition of the newsletter, we have many articles that we hope are of value to you. Some highlights are:

- How to turn your VisualAge Generator server programs into DB2 stored procedures on MVS
- How to access OS/390 VSAM files from programs being debugged in the ITF (enhancement in an upcoming fixpak)
- How to exploit VM saved shared segments.

We also have articles that describe how VisualAge Generator customers, Texaco Brasil and IBM Semea, have leveraged various aspects of VisualAge Generator to their advantage.

The Year in Review

This year has been a very busy one for the VisualAge Generator team. We started the year having just shipped the V3 Refresh in 12/97. Since then, we have shipped a significant number of new capabilities, including:

- V3.0, Fixpak 2 incorporated VisualAge Generator Templates (VAGT) into the base of VisualAge Generator. We also added a set of Wizards on top of VAGT that supports “extreme RAD” creation of the core part of a business system.
- V3.1 included many enhancements, such as Java Gateway support on OS/390 and AIX, native Oracle support, ODBC support on HP-UX, single-system testing of VisualAge Generator Server programs called by non-VisualAge Generator clients, built-in math functions and many usability and performance enhancements.
- V3.1, Fixpak 1 included enhancements to the References utility, the VisualAge Organizer and the VisualAge Generator Parts Browser to make it easier to search for parts.
- V3.1 has recently been verified to support client execution on Windows 98.
- V3.1, Fixpak 2 (soon to be available) will include Euro currency support, ITF access to OS/390 VSAM files.

As always, if you have articles that you would like to share, please submit them to gators@us.ibm.com or fax them to (919) 254-0206.

While the enhancements shipped in 1998 are substantial, we also spent a considerable amount of effort preparing for the future. We ran several “incubator” projects for the line items that will represent the core of our next version. Themes for the next version, which you will be reading more about in future newsletters, should include:

- Easing the transition to e-business
- Easing the transition to Object Technology
- Continuing our strong support for Enterprise RAD
- Extending to more platforms.

Note: IBM plans are subject to change (especially because of the Y2K uncertainties).

In 1998, the VisualAge Generator team participated in several very well-attended customer meetings. Of particular note, the VisualAge Generator Great Lakes Users Group meeting was held at our site in RTP, NC and drew attendance from many companies across North America. Because the meeting was local to the laboratory, many of the VisualAge Generator team members were able to meet with you and discuss ideas. The EMEA VisualAge Generator Symposium was held at the IBM facility in Stuttgart. Attendance was so good that we had to split the agenda and run a parallel track in a second large room. These are the problems we LIKE to have!

To improve customer-to-customer interactions, as a result of your suggestion, we have created an Internet newsgroup for VisualAge Generator. Thus far, the participation has been great. Check out the details and join us at:

http://www.software.ibm.com/ad visgen

I am VERY PLEASED with the progress we made in 1998, and I am EXTREMELY ENTHUSIASTIC about the prospects for 1999. ■

Texaco Brasil

by Luciana Beltrame, Software Account Manager—IBM Brasil and Sue Royer, Worldwide VisualAge Generator Sales

In Brazil, the government-owned oil company is responsible for oil exploration and refining; the real challenge is in the area of distribution and sales. Brazil is the fourth-largest country in the world, so there is a lot of ground to cover. Texaco Brasil uses VisualAge Generator and DB2 to reach across the country.

Background

Texaco Brasil had invested in CSP and SQL/DS in the 1980s to develop applications for their VM host system. These applications, which include accounts payable, inventory control, dealer loans, budgeting and marketing, were still valuable, although not ready for Year 2000, Texaco wanted to protect its investment.

However, some staff members wanted to use Software AG's Natural. In addition, Texaco had an international contract with Oracle, which allowed Texaco Brasil to obtain Oracle application development and database products at no cost. The stakes were high, as IBM and Premier BESTeam member ACE Informatica tried to win the business.

Texaco Brasil evaluated Natural, Oracle and VisualAge Generator with DB2. IBM won the job for two key reasons, according to Daniel Arany, Manager of IS "With VisualAge Generator, we get the ability to develop applications on a single platform and then generate for multiple targets. VisualAge Generator and DB2 together enables us to integrate our applications and use our DB2 data across multiple platforms. We also appreciate that VisualAge Generator is translated into Brazilian Portuguese. This makes it easier for our staff."

Texaco not only had the VM host applications, they also had distributed order processing and billing applications running all over the country. The distributed applications, written in Visual Basic and running on DOS, were difficult to maintain and needed major enhancements. J. Amorim Neto, the CIO of Texaco Brasil, said, "One of the reasons we chose IBM was that we had confidence in IBM's ability to meet our requirements today, and to help us make better use of our technology in the future. VisualAge Generator and DB2 provide us real benefits across all of our platforms."

The Solution

Today, Texaco Brasil has completed the migration of all six major systems comprising almost 900 CSP applications and has made them ready for Y2K. Development and testing is done on OS/2, and runtime is done on VM using VisualGen Host Services, with access to SQL/DS. The solution also includes PComm, which proved to be a better choice because of its technology and price than other emulators the customer was using.

In addition, Texaco is using VisualAge Generator to create new systems to replace the antiquated order processing and invoicing systems. These applications run on Windows NT in 80 locations. Each application uploads sales information from local gas stations to one of five regional AIX systems running VisualAge Generator and DB2 Universal Database (DB2 UDB). The five regional systems communicate with the central VM system using DRDA between DB2 UDB and SQL/DS. If a communications link failure occur between the gas stations running on NT and the regional hubs, the local offices can continue to operate with the invoicing system, which is running stand-alone on the NT workstations, until the link is restored. VisualAge Generator enabled Texaco to rapidly develop the application one time, then generate the code for both Windows NT and AIX.

ACE Informatica played a very key role in the win and in the successful implementation of VisualAge Generator and DB2 at Texaco. ACE also assisted in the evaluation phase, and provided VisualAge Generator education and mentoring to the Texaco development staff. ACE's work was supplemented by the VisualAge Generator Development Services team, which provided configuration assistance on-site.

Many of the end users at Texaco still use 3270 devices. With VisualAge Generator, Texaco can create Text User Interface (TUI) applications for those users, as well as Graphical User Interface applications. The support in VisualAge Generator for creating Web Browser applications and the ability to provide Java client access to all the VisualAge Generator server environments, including VM, were important to Texaco. Daniel Arany states, "Even though we are not using all of its capabilities today, we chose VisualAge Generator as our strategic development tool because we believe it will both protect our investment and take us into the future." ■

A Customer Success Story

IBM Delivers Successful VisualAge Generator Applications

by Stefano Sergi, Angela Carella, Leonardo Degiglio, Flaviano Fiorino, Luigi Glorioso—
IBM Semea SISL Development Team

Who Are We?

IBM Semea Sud is a department of IBM Italia SpA located in the beautiful south of Italy. The mission of this unit is to develop software systems specifically designed to fullfill the requirements of the Italian market. In the past year, a dedicated team of specialists has been involved in the development of SISL2000, a new solution for the Italian health care industry.

Background

The obsolescence of SISL (Sistema Informativo Sanitario Locale), a mainframe-based Information System designed to support the decentralized and local administration of health care providers thorughout Italy, gave impetus to a total rewrite that could incorporate the latest changes in the Health Care Management legislation and at the same time exploit the power of a client/server system architecture: the SISL2000 project was born!

The original SISL system, developed in IBM's Cross System Product working with the DB2 relational database, allowed the integrated management of the typical operations of a hospital or similar health care institute (registration, emergency room management, booking and acceptance of clinical services, clinical history, etc.).

From the time of the initial delivery of the SISL application, a number of changes that occurred in the Italian health care system combined with the transformation and evolution of the Information Technology industry dramatically highlighted several limitations of the existing system.

The mainframe architecture of the system translated into high costs of development, and therefore into high price of the solution, and imposed high costs of system management and maintenance: this severely limited the adaptation of the SISL solution to the needs of small to medium firms, thus limiting its market.

Another serious inhibitor to the acceptance of SISL with new customers was the inadequate text-based user interface, which affected the system usability and was perceived as old technology. In addition, it was quite difficult to integrate the SISL system with personal productivity PC- or LAN-based applications.

Such limitations not only seriously inhibited the marketability and acceptance of SISL with new customers, but also gave way to concerns and dissatisfaction with the existing customer base.

The Action Plan

Once we realized that it was imperative to develop a new, low-cost solution matching the new market demands, in 1996 we began a study to identify the Application Development platform that would allow us to meet an ambitious set of goals.

First and foremost, the existing database design had to remain unchanged, so the new AD environment had to fully support it. Then the new applications had to coexist peacefully with the existing ones, and we wanted to maximize the reuse of existing CSP code while being able to develop sophisticated graphical user interfaces for a network-oriented system.

Another key goal was the ability to deploy the new SISL2000 system on a multitude of platforms so that hardware and software constraints would not limit the use of the application to those customers who could meet the prerequisites.

Just as important was the need for a high-productivity tool that would allow us to develop the new system in a very short time, and that would keep the cost of maintenance within reason. Finally, since the old SISL system was very hard to demonstrate at customer sites, one key requirement was to be able to build a PC-based demo system to take with us when visiting prospective clients.

After a short but accurate analysis, we concluded that VisualAge Generator was the only tool in the marketplace that could meet all of the above requirements. The scripting language, compatible with CSP, allowed us to leverage existing skills and to reuse part of the logic of the existing SISL applications: the entire data management and database integration portion of the system could be fully reused with significant reduction of the cost of delivery. In addition, VisualAge Generator integrated tightly with DB2 while at the same time allowed access to other vendors' database systems.

The graphical user interface development facilities of VisualAge Generator were found to be highly flexible and productive, and particularly well suited for developing reusable system components, which was a major advantage for reducing development time and cost.

The VisualAge Generator portability of the same source to a variety of popular execution environments was superior to any other tool in the marketplace, and we found particularly convenient the fact that we could use a LAN-based, low-cost, development workbench for targeting the creation of a system that could scale up to any platform, including mainframes! Finally, development of a PC-based SISL2000 demo was not only possible, but simple and quick.

The SISL2000 System

The SISL2000 system allows health care institutions to manage all aspects of patients' care. These include hospital admission and registration, first aid patient management, management of bedridden patients, day hospital management, booking and scheduling of clinical services, tracking patients' clinical history, government subsidy reimbursement process management and functions and utilities for the integrated management of data.

The system is composed of approximately 300 GUI client programs and about 600 server programs. The client programs were generated to run on both OS/2 and Windows 95, while the server applications were generated for OS/400, OS/2 and AIX, accessing, respectively, relational data on DB2/400, DB2/2 and Informix via DataJoiner. The client/server communication middleware of the three topologies were CA/400, TCP/IP and DCE, respectively.

SISL2000 has been sold and deployed in production in two hospitals. The first hospital includes approximately 100 OS/2 or Windows 95 (Pentium 166 MHz, 32MB) client machines connected, via CA/400 on a TR-LAN, to an AS/400 S20 server accessing data on a DB2/400 database. The second hospital has approximately 20 Windows 95 (Pentium 100 MHz, 32MB) client machines connected, via TCP/IP on a LAN, to a RISC/6000 server running an Informix database, which is accessed by the server program through the IBM DataJoiner product. This second hospital was also a previous SISL user, and required the implementation of a gradual migration strategy; therefore, the SISL2000 applications have been deployed and coexist with the pre-existing system.

For demonstration purposes we also created a fully functional demo/test/education environment on a portable OS/2 PC, running as a stand-alone architecture using DB2/2.

The design principle of the SISL system is based on the separation of the programming logic responsible for the user dialogs from the program specifications that implement the database access. The GUI applications invoke VisualAge Generator callable functions in a two-tier client/server design model. In this architecture, each server application constitutes a separate logical unit of work and it is the only program aware of the database and authorized to access it. This design improves security, performance and data integrity by preventing uncontrolled direct access to the database from the clients.

Project Results

The SISL2000 project has been a great success. Not only were we able to reuse between 50 and 60 percent of the existing SISL code, but we have achieved all the major objectives that have allowed us to enter the market with a competitive and adaptable solution.

After an initial period of learning and prototyping, we found the visual programming techniques offered by VisualAge Generator to be extremely productive. The average development time of complex GUIs and their respective server programs never exceeded 4 or 5 working days, thanks in particular to the Interactive Test Facility and to the ability to encapsulate complexity inside reusable components.

Deployment of the system on totally different client/server environments did not require us to become familiar with different programming languages and with different system APIs. Except for minor font, color and image problems, the only differences between systems we had to deal with were relative to setting up the runtime environments: a minor investment when compared to the development effort!

Thanks to the RAD development and test VisualAge Generator workbench, which enables us to identify and rapidly fix system anomalies, we are also confident that the maintenance costs of the system will be well below expectations. ■

The 2nd VisualAge Generator International Symposium

On September 8th and 9th, VisualAge Generator held its 2nd International Symposium in Stuttgart, Germany. Over 150 customers and business partners from 18 countries overflowed the IBM Forum conference rooms. The attendance was so great, IBM ran the two-day session split in two rooms and the speakers were busy running from room to room. When they came together for the joint sessions, people had to stand out in the halls!

This annual event gives VisualAge Generator customers and business partners an opportunity to hear about IBM's latest plans and to provide suggestions. This was a very successful symposium, and we hope the next one will be even better.

Accessing OS/390 VSAM Files from ITF and C++ Generated Programs

by Chuck Proffer and Joseph Allen, VisualAge Generator Development and Susan Lafera, VisualAge Generator Information Development

Do you have VSAM files residing on OS/390 that you want to access from a VisualAge Generator program on your workstation? Now, when you test your program using ITF, you can use the same VSAM file you will use in your production environment. This feature is available for ITF with VisualAge Generator Version 3.1 Fixpak 2. Your C++ generated programs already have this capability (since Version 2.2).

Software Prerequisites

System	Local VSAM	Remote VSAM
OS/2	Shipped with IBM VisualAge Generator	IBM Personal Communications AS/400 and 3270 Version 4.11 (or later) IBM VisualAge for COBOL
Windows NT	Not Available	IBM Personal Communications AS/400 and 3270 Version 4.11 (or later) IBM VisualAge for COBOL
OS/390		IBM DFSMS/MVS Version 1.2 (or later)

Setup Required to Access Remote VSAM Files on OS/390

After installing the required products, configuring APPC and verifying that you can establish an APPC session with your OS/390 host, you need to start DFM on the workstation.

Setup for OS/2

IBM VisualAge for COBOL for OS/2 ships a command file to start DFM (startdfm) and a sample configuration file (config.dfm). Modify the startdfm command file to indicate the drive letter that should be used for DFM. The configuration file contains things like USERID, LOCAL_LU, REMOTE_LU and the OS/390 target system name. You must modify the sample configuration file based on your system. After you start DFM, any file name prefaced with the DFM drive letter is assumed to reside on your OS/390 host. Refer to the COBOL documentation for more information on configuring DFM on OS/2.

Setup for Windows NT

IBM VisualAge for COBOL for Windows NT ships a command (dfmlogon) to associate the user's remote user ID and password with a server system and a sample configuration file. You must modify the sample configuration file based on your system. DFM on Windows NT uses a technique similar to the Windows Universal Naming Convention (UNC) to specify the remote file name. The remote file name consists of two back slashes (\\) followed by either the machine name for the remote server or a shortcut name, a single back slash (\), and then the actual file name.

Refer to the COBOL documentation for more information on remote file names, shortcut names and configuring DFM on Windows NT.

Accessing VSAM Files from ITF

To specify that you want to use VSAM files, select the **Options** menu on the **VisualAge Organizer** window. Select **Preferences** and the VisualAge Preferences notebook displays. Select the **VAGen—Test General** tab. At the bottom of the page, select either the **Local VSAM** or **Remote VSAM** radio button and click **OK.** This will cause ITF to use VSAM files for all file accesses (on Windows NT, there is only remote VSAM file support). In a later Fixpak, this option will be moved to the Resource Association Editor so that the type of file accessed can be specified on a file basis. If you have changed your preferences to use **Remote VSAM** and you don't have the communications software setup and working, you will receive an error.

In addition to changing your preferences, you also need to specify the physical name and path in the ITF Resource Association File editor. In the **Physical name** field, specify the file name as it is on your OS/390 system but without the high-level qualifier. If the file does not already exist on your OS/390 system, VisualAge Generator will create it for you. On OS/2, in the **Path** field, specify the machine name or a shortcut name using the Universal Naming Convention.

Accessing VSAM Files from C++ Generated Programs

Access to VSAM files from a C++ generated program is determined by the resource association file (RSC). Specify /FILETYPE=VSAM in the ASSOCIATE entry for a VSAM file. Remember that there is no local VSAM support on Windows NT, so all VSAM file access is remote. If you do not have the communications software set up and working, you will get an error. To access a remote VSAM file on OS/2, preface the file name with the DFM drive letter. On Windows NT, specify the file name using the Universal Naming Convention. Refer to the *VisualAge Generator Server Guide for OS/2, Windows NT, HP-UX, and AIX* for more information on using VSAM and resource association files.

Diagnosing Error Conditions When Using VSAM with ITF

A trace facility has been provided to assist in diagnosing error conditions. The trace is controlled by the HPTTROPT environment variable. Specifying HPTTROPT=1 turns on the trace, and specifying HPTTROPT=0 turns off the trace. The trace output is written to a file named hpttrace.out unless you change the name using the HPTTROUT environment variable.

Diagnosing Error Conditions When Using VSAM with C++ Generated Programs

The trace facility for C++ generated programs is controlled by the FCWTROPT environment varaible. Specifying FCWTROPT=31 will turn on trace for file I/O as well as other C++ program-related events. The trace output is written to a file named fcwtrace.out unless you change the name using the FCWTROUT environment variable. Refer to the appendix in the *VisualAge Generator Server Guide for OS/2, Windows NT, HP-UX, and AIX* for more information on the trace environment variables. ■

VisualAge Generator Web Pages

The VisualAge Generator web address is:
www.software.ibm.com/ad/visgen

For IBM's predecessor 4GL, Cross System Product, the web address is:
www.software.ibm.com/ad/visgen/csp

Running VisualAge Generator Programs in a VM Saved Shared Segment

by Debbie Prevost, VisualAge Generator Information, and Bert Paul, David Weiss and Blaine Laufmann, VisualAge Generator Development

Note: This article was originally published in the October 1998 issue of VisualAge Magazine. You can find out more about the magazine at web site http://www.vamagazine.com.

A new feature of VisualAge Generator Server for MVS, VSE and VM Version 1.2 (released in June 1998) enables you to run programs generated with VisualAge Generator from a VM saved shared segment. A saved shared segment is a range of pages in virtual storage that you can define to hold data and programs with reentrant code. The saved shared segment is a public space that can be shared by any authorized virtual machines on a VM system.

Benefits

You can gain the following benefits from using the VM saved shared segments feature of VisualAge Generator Server:

- You can avoid the problem of running out of resources during program execution, because the shared space can be much larger than the typical storage and memory allocated for an individual user ID. Saved shared segments attached to a virtual machine can reside above that virtual machine's defined storage, and free the individual user ID's storage for other purposes.
- Swapping is reduced when programs are stored in a saved shared segment. Using saved segments decreases the I/O rate and DASD paging space requirements, and improves virtual machine performance.
- Overall storage use can be reduced, because several users can access the same physical storage.

Restrictions

Generated programs run from saved shared segments cannot use the XCTL command to transfer control to either a VisualAge Generator generated program (using XFER) or a non-VisualAge Generator generated program (using DXFR).

Example

This example is designed so you can use it as a blueprint for:

- Generating programs that will be run from a VM saved shared segment
- Setting up a saved shared segment and storing your VisualAge Generator-generated programs in it
- Running your programs from the saved shared segment.

In the example, a program named A13MP01 performs a DXFR call to program A13MP02.

A13MP01 displays a menu panel, and A13MP02 is a secondary screen accessed through the A13MP01 menu.

A13MP01 contains no SQL processing, and no file I/O is generated. For a program using either SQL or file I/O, more setup steps would be required. Refer to the *VM/ESA Planning and Administration* manual for more information.

While some of the steps in the example are performed by the VisualAge Generator programmer, other steps are done by the system administrator. Each set of steps is labeled according to who should complete those actions.

Try the Example Yourself

If you want to try the example on your system, you can download the example programs from this web page:

http://www.software.ibm.com/ad/visgen/downloads

From this web page, click on the download link for the VisualAge Magazine example, and download all the files on the subsequent page.

After you download the files, you will need to import A13MP01.ESF and A13MP02.ESF into an existing program on your system. The options file (VMCMS1.OPT) contains example code you can use to modify your own options file.

Programmer's Task—Generate Programs

To generate the programs, the programmer should do the following:

1. Generate program A13MP01, using the VisualAge Generator defaults.
2. Edit the generated and uploaded A13MP01 EXECP to change the erase_text_option from Y to N.
3. Issue the following command:

   ```
   PIPE REXX (A13MP01 EXECP A)
   ```

   This command recompiles A13MP01 and leaves the file A13MP01 TEXT on your A disk.
4. Verify that generated file A13MP01 TEXT is on your A disk.

5. Copy and rename the following files:

Copy File	Rename Copy To	Description
G13MP1FM EZEFOBJ A	G13MP1FM TEXT A	Map group
HLPLI1FM EZEFOBJ A	HLPLI1FM TEXT A	Help map group
TMAPOPS EZEFOBJ A	TMAPOPS TEXT A	Valid menu options table

These renamed text decks and A13MP01 TEXT make up the full function of A13MP01.

6. Erase the generated EZEAPPL LOADLIB file on your A disk to prevent accidental access of A13MP01 from your A disk loadlib and ensure the program is accessed from the saved shared segment.

Caution: If you have information about other programs stored in EZEAPPL LOADLIB, do not erase this file. Instead, rename it so it will not be accessed during this example.

7. Generate program A13MP02, using the VisualAge Generator defaults.

Administrator's Tasks—Set Up Saved Shared Segment and Store Programs

The administrator should perform the following tasks:

1. Ensure that your administrator ID has authority to issue CP class E commands and your virtual machine storage is large enough to allow you to define and generate a physical segment outside the normal memory range for user IDs.

2. Issue the command ELASETUP to link to the disk with SELALKED TXTLIB.

3. Use the #CP LINK and ACC commands to link to the disk that contains SCEELKED TXTLIB.

4. Issue the following command to link the disk containing SELALKED with the disk containing SCEELKED:

```
GLOBAL TEXTLIB SELALKED VMLIB
SCEELKED
```

In this example, the disk containing the VMLIB TXTLIB resides on the CMS S disk.

5. Use XEDIT to create a physical segment definition file named PSEGBERT PSEG that contains these lines:

```
LSEGMENT A13MP01  LSEG
LSEGMENT G13MP1FM LSEG
LSEGMENT HLPLI1FM LSEG
LSEGMENT TMAPOPS  LSEG
```

Each line in the PSEGBERT file points to a logical segment used by the A13MP01 program.

6. Use XEDIT to create a logical segment definition file for each of the logical segments listed in PSEGBERT PSEG. In the following example, each file contains one line:

Logical Segment Definition File	File Contents
A13MP01 LSEG	TEXT A13MP01
G13MP1FM LSEG	TEXT G13MP1FM
HLPLI1FM LSEG	TEXT HLPLI1FM
TMAPOPS LSEG	TEXT TMAPOPS

Refer to the chapter on planning and defining CMS logical saved segments in the *VM/ESA Planning and Administration* manual for a description of physical and logical saved segments.

7. Issue a DEFSEG command in the following format to reserve space for the saved shared segment:

```
DEFSEG (pseg_name) (range) SR
```

In our example, we used an address range of 2900000 to 29FF000 by issuing the command:

```
DEFSEG PSEGBERT 2900-29FF SR
```

Where:

PSEGBERT Is the name of the saved shared segment. You can use any name.

2900-29FF Is the page range reserved for the saved shared segment.

SR Specifies that this segment is shared (S) and read-only (R).

8. Issue the following command to generate the physical segment:

```
SEGGEN PSEGBERT
```

The above command creates six files on your A disk:

- A13MP01 LSEGMAP
- G13MP1FM LSEGMAP
- HLPLI1FM LSEGMAP
- TMAPOPS LSEGMAP
- PSEGBERT PSEGMAP
- SYSTEM SEGID.

These LSEGMAP and PSEGMAP files should resemble the examples printed in the *VM/ESA Planning and Administration* manual. If the SEGGEN command fails, the LSEGMAP files might be created, but they will contain only error messages, and the SYSTEM SEGID file will not be created.

For a description of the SEGGEN command, see the *VM/ESA CMS Command Reference.*

9. Replace the file on your administrator's 190 disk with SYSTEM SEGID and "resave" CMS. The procedure for resaving CMS is described in the *VM/ESA Planning and Administration* manual.

10. Inform the programmers that the saved shared segment is ready for use.

Programmer's Tasks—Prepare User ID and Run the Program

1. While the system administrator is resaving CMS, take the following steps on your own A disk:

  a. Copy ELARUN EXEC to your A disk and rename it to ELARUNSS EXEC.

  b. Edit ELARUNSS EXEC A as follows:

    1. Comment out or delete the line *'OSRUN 'parm1.*
    2. Replace the deleted line with this coding:

```
do;
address 'COMMAND'
'SEGMENT LOAD' parm1
address 'COMMAND'
parm1
end;
```

Note: This code is part of an Else clause and must come before the line ApplRetcode = RC.

    3. Copy A13MP01 EXECX A to A13MP01 EXEC A. This exec is used to execute A13MP01.
    4. Edit A13MP01 EXEC and change the line containing ELARUN to point to ELARUNSS instead.

Note: You might need to further modify A13MP01 EXEC for it to run on your system. For example, you might need to change some access paths or dynamic accessing file names.

2. After the system administrator has resaved CMS, ensure that the virtual storage for your user ID is less than the address range defined for the saved shared segment. If needed, redefine your A disk's virtual storage. For example, if the saved shared segment uses the range 29M–36M, you might want to redefine your virtual storage to 24M, using these commands:

```
DEF STOR 24M
IPL CMS
```

3. Enter A13MP01 to run the program. In this example, A13MP01, which resides in a saved shared segment, performs a DXFR call to A13MP02, which resides on your A disk instead of in the saved shared segment.

4. Exit the program and return to a CMS Ready prompt.

5. Issue the following command:

```
QUERY SEGMENT
```

The output of this command shows that A13MP01, G13MP1FM and HLPLI1FM are listed with the other system and product programs that are loaded above the 29M line. Thus, these programs did not get loaded until you needed them during execution.

Additional Information

See the following VM/ESA version 2.2 books for more information on saved shared segments. It is highly recommended that you refer to these books before generating programs in saved shared segments.

- VM/ESA Planning and Administration
- VM/ESA CMS Application Development Guide
- VM/ESA CMS Command Reference
- CP Command and Utility Reference.

■

Correction

The previous issue of the newsletter stated that Sue Royer was the author of the article on Nikon Optical. This was incorrect; Mike Wu, VisualAge Marketing, was the author. My apology to Mike for this error.

Developing Stored Procedures for DB2/MVS

by Roger Newton, Paul Hoffman, and Denise Hendricks, VisualAge Generator Development

Transforming new or existing VisualAge Generator client/server programs into DB2 stored procedures for DB2/MVS can be done with little effort. The following sample programs and related files have been created to guide you through this process. The source code for the sample files are included in the self-exploding file, STORPROC.exe, which can be obtained from the VisualAge Generator ftp site at URL:
ftp://ps.software.ibm.com/ps/products/visualagegen/info/v3.1

The files included in the STORPROC.exe file are:

STORPROC.htm Instructions for defining a VisualAge Generator stored procedure (in html format).

STFLIST.esf The stored procedure to be executed on DB2/MVS

ASTPROC.esf The calling program that invokes the stored procedure

STAFF.qmf STAFF table in QMF format. To upload the file to the host using PCOMM, issue the following command:

```
send staff.qmf b:qmf.staff
  [crlf recfm(f) lrecl(44)
```

STFPROC.sql SQL statements used to catalog the stored procedure in the DB2 system catalogs.

To unzip the files, issue STORPROC on the command prompt.

Defining Stored Procedures

A stored procedure is defined as a called batch application that accesses only relational or DL/I databases, not files. The stored procedure cannot XFER to another program. The only other special considerations for defining stored procedures are in defining the parameter list. The following is a list of the special considerations for defining the parameter list for a stored procedure:

- Parameters can be individual data items up to 254 bytes long or records up to 32K bytes long.
- Individual data item types are restricted to the types supported by SQL row records: CHA, DBCS, BIN, PACK and HEX.
- Record definitions must be defined with a top-level character item that includes all of the other items in the record. The top-level item is specified as the parameter in the SQL CALL coded in the calling program.
- Records up to 254 bytes long can be passed as fixed-length SQL CHAR parameters.
- Records greater than 254 bytes in length must be passed as SQL VARCHAR parameters, and a second record definition must be defined for use in the called parameter list with a 2-byte BIN data item added to the front of the record structure.

Defining the Stored Procedure Call

To define the stored procedure call, do the following:

1. Use an SQLEXEC process to define the SQL CALL statement to call the stored procedure, specifying the individual data item or top-level record item for each parameter as a host variable on the SQL CALL.
2. Since DB2 does not know about the structure of record parameters, call EZECONV for each record parameter before the call to convert the parameter to host data format on the way to the stored procedure, and after the call to convert the parameter back to client format.
3. Specify individual items or top-level record items of up to 254 bytes long as host variables on the SQL CALL statement.
4. For records greater than 254 bytes, define an SQL record item with the same length as the record and with SQL data code = 457 (VARCHAR). Specify the SQL item as the host variable and move the record contents from the record to host variable prior to the call, and back from the host variable to the record after the call.

Preparing a VisualAge Generator Stored Procedure

You prepare a VisualAge Generator stored procedure as you would prepare any other MVS BATCH called program. The program should be link-edited with the Call Attachment Facility language interface module, DSNALI. The template, EFK2MPBC.TPL (shown below), was permanently changed to use the CAF interface module.

EFK2MPBC.TPL

```
//*********************************************************************
//** EFK2MPBC - PREPARE MVSBATCH APPLICATION WITH DB2 ACCESS
//** AND NO DLI ACCESS
//** DB2 PRECOMPILE, COMPILE, LINK AND BIND
//*********************************************************************
//PCLB EXEC ELAPCLB,MBR=%EZEMBR%,ENV=%EZEENV%,DATA=%EZEDATA%,
//CGHLQ='%EZEPID%'
//L.SYSIN DD *
CHANGE ELAAPPL(%EZEMBR%)
INCLUDE SELALMD(ELARMAIN)
INCLUDE SELALMD(ELARSINT)
INCLUDE SELALMD(ELASTB07)
INCLUDE SYSLIB(DSNALI)
ENTRY %EZEENTRY%
NAME %EZEMBR%(R)
/*
//B.SYSTSIN DD DISP=SHR DSN=%EZEPID%.%EZEENV%.EZEBIND(%EZEMBR%)
```

Declaring Stored Procedures

All stored procedures have to be defined in the DB2 system table SYSIBM.SYSPROCEDURES. The following information was cataloged for this example of a VisualAge Generator stored procedure.

Note that the stored procedure name, STFPROC, is different from the load module name, STFLIST.

```
INSERT INTO SYSIBM.SYSPROCEDURES
(PROCEDURE, AUTHID, LUNAME, LOADMOD, COLLID, LINKAGE,
LANGUAGE, RUNOPTS, STAYRESIDENT, IBMREQD,
PARMLIST)
VALUES('STFPROC', ' ', ' ', 'STFLIST', 'STFLIST',' ',
'COBOL',' ', 'Y', 'N',
'STFPARM SMALLINT INOUT,
VARCHAR(1660) FOR BIT DATA INOUT')
```

Parameter List Data Types

The parameters passed by DB2 to the stored procedures can be in several forms (using SQLDA, host variables, constants, and NULL). Currently, VisualAge Generator stored procedures support passing parameters only as host variables. The host variables for records and hex data items should be defined as binary data (FOR BIT DATA) when defining the parameters in the DB2 system catalog. This prevents DB2 from performing conversion on the data being passed. Refer to the section "Declaring Stored Procedures" for an example on how to define the parameters.

Parameter Size

When calculating the size of a VARCHAR parameter, do not include the length of the additional 2-byte field in the size. You will notice that the parameter STFLIST_REQ_MSG was defined with a length of 1660 to DB2. The extra 2 bytes will be inserted by DB2 when the parameter is passed up.

DB2 Linkage Conventions

Parameters for VisualAge Generator stored procedures should be defined as INOUT (input/output) parameters. *INOUT* means that data will be flowing to and from the stored procedure. The sample called program was tested with both SIMPLE and SIMPLE WITH NULLS. The SIMPLE linkage convention was the only parameter list convention that worked successfully for VisualAge Generator. To specify the SIMPLE linkage convention, enter a blank value for the LINKAGE column when the stored procedure is defined to DB2.

Binding the Stored Procedure Package

The following bind command was used to create the package for the sample VisualAge Generator stored procedure, STFLIST. If your program calls other generated programs using the VisualAge Generator CALL or DXFR statement, include the DBRMs for these programs in the stored procedure package.

```
DSN SYSTEM(DSNE)
* EFK2MBDD
* BIND TSO APPLICATION WITH DB2 ACCESS AND NO DLI ACCESS
* BIND MVSBATCH APPLICATION WITH DB2 ACCESS AND NO DLI ACCESS
BIND PACKAGE(STFLIST) -
MEMBER(STFLIST) -
ACT(REP) -
VALIDATE(BIND) -
ISOLATION(CS)
* OWNER(OWNERGRP)
BIND PACKAGE(STFLIST) -
MEMBER(ELADBRM4) -
ACT(REP) -
VALIDATE(BIND) -
ISOLATION(CS)
* OWNER(OWNERGRP)
```

Calling a VisualAge Generator Stored Procedure

Stored procedures are invoked using the SQL statement CALL. The program, ASTPROC, contains an example of how to call a stored procedure. The program was generated to run as a C++ program in the Windows NT environment. The steps to prepare ASTPROC to call stored procedures, STFLIST, are discussed in more detail below.

Host Variables

1. Create host variables for each parameter passed to the stored procedure.
2. Define host variables corresponding to working storage records greater than 254 bytes long as VARCHAR, data code 457.
3. Use the default SQL data code for the item type for host variables that are less than or equal to 254 bytes long.

For our sample, two host variables that correspond to the two parameters were created, TEST_REC.STARTING_ID and TEST_REC.STFLIST_DATA. Both variables were defined in the same SQL row record. We could have defined an SQL row record for each parameter.

Invoking the Stored Procedure

Invoke the stored procedure (as shown below) from a SQLEXEC process, passing the host variables in the order expected by the called program.

```
CALL  STFPROC
 (:TEST_REC.STARTING_ID,
  :TEST_REC.STFLIST_DATA)
```

Data Conversion

Because DB2 does not understand how to work with VisualAge Generator data structures, the record parameter is defined as bit data to DB2. As a result, the calling program is responsible for converting the data before and after calling the stored procedure. The following is an example of how this was coded in the main process, PSTPROC_MAIN.

```
EZEFEC = 1;
PERFORM PSHOW_MSTFMN;
WHILE EZEAID NOT PF3;
STFLIST_SEARCH_ROW.STARTING_ID = MSTFMN.STARTING_ID;
TEST_REC.STARTING_ID = STFLIST_SEARCH_ROW.STARTING_ID;
CALL EZECONV STFLIST_REQ_MSG2,'L','ELACNENU';
PERFORM PCALL_STORED_PROC;
; /* CALL STFLIST  STFLIST_SEARCH_ROW, STFLIST_REQ_MSG2;
IF EZESQCOD NE 0;
MOVE EZESQCOD TO MSTFMN.STAFFIDX_WS(1);
CALL EZEROLLB;
END;
STFLIST_REQ_MSG2.STFLIST_DATA = TEST_REC.STFLIST_DATA;
CALL EZECONV STFLIST_REQ_MSG2,'R','ELACNENU';
MOVEA STFLIST_REQ_MSG2.STAFFIDX_WS TO MSTFMN.STAFFIDX_WS FOR 10;
MOVEA STFLIST_REQ_MSG2.NAME_WS TO MSTFMN.NAME_WS FOR 10;
MOVEA STFLIST_REQ_MSG2.SALARY_WS TO MSTFMN.SALARY_WS FOR 10;
MOVEA STFLIST_REQ_MSG2.COMM_WS TO MSTFMN.COMM_WS FOR 10;
PERFORM PSHOW_MSTFMN;
END;
```

Testing Stored Procedures

The dynamic SQL interface used by the Test Facility does not support calling a stored procedure. The best way to test a stored procedure is to test it as a normal client/server program using the CALL statement. The following list describes what we did to test our stored procedure:

- Commented out the SQLEXEC process containing the DB2 SQL statement CALL and replaced it with a VisualAge Generator CALL
- Commented out the calls to the conversion routines
- Commented out the statements that moved the data back and forth between the working storage records and the host variables
- Removed the VARCHAR length variable from the parameters on the called parameter list.

When you are ready to generate the called program for MVS BATCH, reverse the steps above. Be sure to insert the VARCHAR length variable back into the appropriate working storage record. For this example, record STFLIST_REQ_MSG was the only record that required the VARCHAR length variable.

After the stored procedure has been prepared and ready to start, the calling program can be generated and tested.

Tracing and Debugging

To debug your VisualAge Generator client/server program, set the environment variable CSO_DUMP_CONV=YES to show the conversion of data before and after the call to the stored procedure as follows:

```
SET  CSO_DUMP_CONV=YES
```
■

Gathering the Necessary Documentation for Problem Reports

by Jim Eberwein and Jim McVea, VisualAge Generator Support Team

In previous issues of the VisualAge Generator Newsletter, we have documented processes you can follow to diagnose some of the problems you might have encountered executing your VisualAge Generator programs. Unfortunately, there might come a time when these processes do not provide enough information for you to diagnose the problem. Your only option at this time might be to open a problem record (PMR) with the IBM Support Center. Appendix D of the *VisualAge Generator Installation Guide* contains instructions for opening a PMR via telephone support.

Once the PMR has been routed to the VisualAge Generator Support Team, the support representative might request specific product documentation to assist in problem diagnosis. The August 1997 edition (Vol. 2, No. 3) of the VisualAge Generator newsletter contained the article "Tracking Down Those Bugs," which documented the steps necessary to produce trace information. In addition to this trace information, additional documentation might be requested that will be used to diagnose the reported problem. The goal of this article is to explain how to produce the requested documentation.

When a problem is reported to the Support Center and is not a known defect, one of the first goals of the support team is to try and re-create the problem. Unless you have already determined the exact cause of the problem, the support representative might request your program definition, data files and environment information. The following examples illustrate how to perform the commands to generate the requested documentation:

Program Definitions

Program definitions can be forwarded to IBM via the following VisualAge utilities:

VisualAge Smalltalk "Export Applications"

GUI class definitions are no longer considered VisualAge Generator members. If the problem you are experiencing uses GUI classes, you must use the VisualAge Smalltalk Export utility to transfer your GUI class definitions to IBM. To export your GUI class definitions, do the following:

1. Change the name of the application manager and the name of the class owners to "Library Supervisor."
2. Version the application.
3. Select the application you want to export from the **VisualAge Organizer** window.
4. From the **Applications** pull-down menu, select **Import/Export,** then select **Export.**
5. When prompted for a library name, enter the name of a *.dat* file you would like the versioned application to be copied to. By default, all subapplication versions are copied to the library file. If the application has prerequisites, you will need to export these applications as well.

VisualAge Generator "Export"

Depending on the number of applications you need to export, it might be easier to use the VisualAge Generator *Export* command to create an ESF file that contains the definitions for the VisualAge Generator members. If this techique is used, care must be taken to ensure that the member's associates are also exported. To expand a program and its associates, do the following:

1. From the **VisualAge Organizer** window, select the VisualAge Generator program part that you want to export. This part is in the **Organizer's** third pane.
2. From the **VAGen Parts** pull-down menu, select **Associates.**
3. On the **Associates** window, choose the **View** pull-down menu and select **Select All.**
4. On the **Associates** window, choose the **Parts** pull-down menu and select **VAGen Export.**
5. When prompted for a file name, enter the file name you want to export the VisualAge Generator members to.

Program Data

When re-creating problems, it is likely that the program data must also be forwarded to IBM. The two common database managers used by customers are DB2 and Oracle. The following describes how to transfer the data used by the programs to IBM.

DB2 Export

To export data from DB2, do the following:

1. Open a DB2 Command prompt.
2. Connect to the database by issuing the following command:
```
CONNECT TO <databasename>
```

3. Issue following command to transfer data and table information to a file:

```
EXPORT TO <filename>.lXF
OF IXF MESSAGES MSGS.TXT
SELECT * FROM <tablename>
```

On completion of the above command, two files are produced. The .IXF file contains the table definition and data. Messages are recorded in the MSGS.TXT file.

Oracle Export

Oracle database export supports only exporting into an Oracle binary format. To export data from Oracle, do the following:

1. Open the Oracle Data Manager and logon to the database.
2. Select the **Export** tab and specify a file name and location for your data.
3. Select the **Specify** button and select the tables that will be required to reproduce your testcase.
4. Click on the **Export** button and Oracle generates the required file.

The above steps exports an entire table and all of its data. If the problem re-creation does not require that the database tables contain data, a data definition language (DDL) file will work. The DDL file will need to be typed manually and it needs to contain only the creation of the table and its columns.

Program Traces

In addition to the traces described in the “Tracking Down those Bugs” article, the support representative might request additional VisualAge traces. This section describes how to produce these additional traces.

Connection Tracing

Sometimes the problem might be that some of your GUI connections are not firing correctly or they are not in the sequence that you expect. In these cases, you can generate a connection trace to examine the connection sequences by doing the following:

1. Open the GUI **Composition Editor.**
2. From the **Options** pull-down menu, select **Debug.** This opens a **Connection Trace Log** window and a test of your GUI begins automatically. A debugger window might also open because, by default, the connection tracing breaks on all connections.
3. Before closing any debuggers, go to the **Connection Trace Log** window and select the **Break On: None** radio button and make sure that the **Trace: All** radio button is selected.
4. If a debugger window opened, press the **Resume** button to continue testing your GUI.
5. Interact with your GUI as you would during a normal test. As each connection is fired, it is written to the **Trace Log** window.
6. After completing your testing, you will need to select all of the text in the **Trace Log** window, copy and paste it into a text document that can be saved and sent to support.

ODBC Tracing

New ODBC tracing capabilities were introduced with VisualAge Generator V3.1. To generate ODBC trace statements from your C++ applications, you need to set the FCWTROPT=159 environment variable. This produces an FCWTRACE.OUT file as described in the “Tracking Down those Bugs” article.

Product Message Information

System Transcript

If you are experiencing an error during an import/export or load/unload, then the messages printed in the System Transcript might prove useful to the support team. You need to select only **Save As...** from the **File** pull-down menu in the **System Transcript** window to save its contents.

Packaged Image Runtime Errors

If you receive an error during execution of a packaged image, the support team needs a copy of the walkback.log (if created) and possibly a copy of the *.es files, which describe the classes and methods packaged with the runtime image. These *.es files can be found in the same directory as the executable file created during the packaging process.

Debugger

If you get a debugger window, the support rep will request the “debugger stack trace text.” This provides the support team with a listing of the Smalltalk classes and methods that might be causing the failure. When the debugger window opens, select **Save Stack Trace Text As...** from the **Stack** pull-down menu, then specify the file name and location of the stack trace file.

Determining Fixpak Level

To determine the fixpak level of VisualAge Generator V3.x, do the following:

1. Go to the **VisualAge Organizer** window and select **Parts Browser** from the **VAGen Parts** pull-down menu.
2. From the **Parts Browser** window, select **Product Information** from the **Help** menu. The build information will be located in the lower left corner of the splash screen.

System Environment Information

One of the first questions the support representative will ask is the name of the operating system on which the error occurred. On some occasions, you might be asked specific questions regarding the system environment. Past experiences have indicated that this information is usually helpful when debugging problems that occur in the AIX environment. In this section, we discuss the information that might be helpful when working with problems that occur under AIX.

AIX Environment

If the reported problem is occurring under AIX, the support representative might request some information regarding the environment setup. The following list details the information that might be requested. Commands to produce this information are also included:

1. The profile for the userid that is logged in:
   ```
   /home/userid/.profile
   ```
   If this is a CICS problem, also get the CICS environment file:
   ```
   /var/cics_regions/
   region_name/environment
   ```
   where *region_name* is the CICS region where the problem is occurring.
2. The current environment settings:
   ```
   set > env.lst
   ```
3. What the file systems look like:
   ```
   df > filesys.lst
   ```
4. A file list of the home directory:
   ```
   Is -la $HOME> home.lst
   ```
5. A file list of the genout directory:
   ```
   Is -la genoutdir > genout.lst
   ```
   where *genoutdir* is the genout directory.
6. A list of the software on the system:
   ```
   Islpp -h all > software.lst
   ```
7. A dump showing the version of Workgroup Services that a .dll was prepped with:
   ```
   dump -H <dllname>> wgsprep.lst
   ```

LE/370 Debugging

LE/370 is the common runtime environment used by the generated COBOL programs running under MVS, VM and VSE. Diagnosing problems under LE is a bit different prior to LE/370, since LE/370 converts traditional system abend codes into U40xx abend codes. If you experience an abend running your COBOL program, check to see if a dump was written to the CEEDUMP dataset.

CEEDUMPs are useful for finding the cause of the abend and the module in which the abend occurred. Sections of the CEEDUMP that are of interest are the "Traceback" and "Condition information for Active Routines."

The "Traceback" section shows the current program stack. Usually the module CEEHDSP is the top module in the "Traceback" section. This module is used by LE to schedule the CEEDUMP. It should be ignored. Instead, look at the status field in the "Traceback" section. This field indicates the condition that resulted in the module giving up control to the previous program in the trace. For abending programs, you will want to see which program experienced the abend. This is indicated by the "Exception" keyword in the status field.

After determining the failing module, you need to determine the type of abend. This information is included in the "Condition information for Active Routines" section. Under the heading "Original Condition:" is a message prefixed by CEE that indicates the type of abend. For common program interruption codes, (ABENDOCy) LE/370 issues message CEE32xx, where *xx* is the hex equivalent of y. For example, an ABEND0C7 is reported as CEE3207.

Using the information from the "Traceback" and "Condition information for Active Routines" sections, the IBM support representative might be able to determine if the problem is a known defect. If it is, you will be informed of the APAR number that is being used to track the problem and if a fix is available. If this is a new problem, the support representative will most likely request a system dump to diagnose the problem. LE/370 has documented instructions for obtaining system dumps in information APAR II10573. If you do not have access to this APAR, the support representative can provide you with a copy of the APAR. ■

The Most Common Migration Mistakes

by Jeri Petersen, VisualAge Generator Consulting Services, and Nina Newton, VisualAge Generator Information Development

When mistakes are made, you can benefit from those mistakes. One of the ways you can take advantage of others' mistakes are through forums. You can read about the problems others have run into and the solutions that they recommend. This article discusses common mistakes made by VisualAge Generator users, when migrating from one version to another. The information for this article was obtained from looking through past forum entries, talking to other consultants in the group and personal experiences working with customers. The list of mistakes is based on when they typically occur. Other than that, there is no particular order.

Installation Mistakes

1. Installing versus Loading

The most common problem related to installation is missing the distinction between install and load. VisualAge Generator is a feature of the VisualAge Smalltalk product. You must install VisualAge Smalltalk, then install VisualAge Generator Developer and finally load the VisualAge Generator Developer feature code into the library manager and the client's image. If you bring up VisualAge Generator Developer and you see only two panes in the **VisualAge Organizer** window, this means the feature is not loaded.

2. Omitting the Prerequisites Installation

Another mistake we run into frequently when doing services is that the customer do not have the proper prerequisites installed. VisualAge Generator Developer requires a minimum of 64MB of memory. This 64MB does not include other products like notes or mail, multiple host sessions, web browsers, or other tools that you might want to run at the same time you run VisualAge Generator. The more memory you have, the better.

For those of you already on VisualAge Generator V2, you might be using LAN generation. REXX was the tool used to implement LAN generation. REXX is part of OS/2, but is not included with Windows NT. Therefore, if you plan to do LAN generation on Windows NT, you need to get IBM Object REXX for each client and each LAN generation machine.

PCOMM is the only tool supported for upload using SNA. However, TCP/IP is the preferred upload technique. It is much faster and more reliable than the SNA upload. However, you do need to install TCP/IP on your host machine.

Mistakes Before Migration

3. Lacking ENVY Education

It is difficult, if not impossible, to plan your migration without having some ENVY education. If you cannot attend a class, we recommend that you read the *IBM Smalltalk User's Guide (SC34-4536)* and the *VisualAge Generator System Development Guide (SG24-4230).*

4. Forgetting to Save an Empty Manager and an Empty Image for Future Use

Before you start storing any of your own code in the library manager, you should do the following:

- Save an empty manager and an empty image.
- Save a copy of the abt.ini file.

The empty image is not really empty, but contains the VisualAge Smalltalk and VisualAge Generator code. You can copy the abt.icx file and the abt.ini file to other developers' machines so that they do not each have to load the VisualAge Generator Developer feature. The empty image also gives you a starting point for building baseline images. For example, if you do a weekly build, you would add your current level of code to the empty image to create the new baseline for the week.

The empty manager is also not really empty, but contains the VisualAge Smalltalk and VisualAge Generator code. You will use the empty manager when you want to reset your manager after your pilot migration and before starting your real migration. You can also use the empty manager when you want to get rid of old code. To do this, just export the most recent versions of your configuration maps and applications to the empty manager.

5. Underestimating the Number of Parts

Underestimating the number of missing, duplicate, and unused parts is another problem. We have not worked with a customer yet who was not surprised by at least one missing or duplicate part. Unused parts do not present a real problem. You can put the unused parts in an "UnusedPartsApp" or not migrate them to ENVY. However, missing parts have to be located and unintended duplicate parts must be resolved. This takes time. In addition, if you discover a lot of missing or duplicate parts, you really need to generate and test to be sure you have found the real source code that matches production.

6. Ignoring Special Considerations

If you are migrating from Cross System Product (CSP) V3.3 or earlier, you must regenerate. Some customers have no serious problems with the generated COBOL. Others have to make some code changes. For example, if you have used a COBOL reserved word as a program name, you must change the program name and all references to it.

If you are migrating from any release of CSP, you will be moving to a workstation development environment. You need to allow time to set up that environment.

For GUIs, you must generate the VisualAge Generator runtime code, package, and test your GUI.

Last but not least of the special considerations (and the only one that saves you time) is the use of HPTRULES.NLS. This is a control file that enables you to specify an alternate not sign that is to be converted to the primary not sign when you use VisualAge Generator import. This means that if the only special character you are having a problem with is the not sign, you can modify HPTRULES.NLS so that you do not have to modify the esf files or your download table.

7. Igoring that ENVY is Different from MSLs

Another mistake is ignoring that ENVY is just different from your MSL environment. Locking into your current MSL process and expecting to carry it over intact is not necessarily the most efficient way of using ENVY. For example, because ENVY automatically creates new editions of the VAGen parts each time you do a save, having a list of associates might be better than having an esf export file. The list of associates tells the VAGen part name, its edition time stamp, and the ENVY application name.

Ownership is much more important in ENVY. Ownership is also a completely new concept. You need to understand what it will do for you and to you so you can plan your development environment and processes accordingly.

Mistakes During Migration

8. Using a Machine That is Too Small

During migration, you want an even bigger machine than what you might have for your typical clients. You want the migration tool to be able to analyze your source code. To do that the migration tool needs as much memory as you can make available. Processor speed is important, but memory seems to be the most critical for migration. 128MB or more is best. One thing that can help is to install VisualAge Generator Developer on the same machine as the VAST Library Manager. Then use the server machine to do your migrations. Typically the server is bigger and faster than most client workstations.

9. Ignoring Ownership Issues

Ownership is very important. Here are some considerations to follow:

- It is recommended that you put just one program into an ENVY application. This is one technique for addressing ownership. It helps to ensure that two developers are not trying to modify the same ENVY application at the same time. However, you should temper based on your own systems.
- If you have common code that is maintained by only one person and is relatively stable, dividing along functional lines within the common code might make more sense.
- If you already have ownership in place, such that one person and only one person maintains a subsystem, then dividing along functional lines rather than one program per application might reduce the number of ENVY applications and simplify the ENVY administration functions.

10. Making Applications Too Large

Trying to size the ENVY applications properly is difficult. However, there are a few general guidelines:

- Do not put 30,000 data items into a single ENVY application. You will not be happy with the performance. You will not be happy with even 3,000 data items in a single application. It is recommended that you put no more than 500 parts of a single type in an application.
- If you have lots of DB2 tables and a set of programs for each table that controls all access to the table, you might want to create one ENVY application per table and include all the programs that control access to that table in that application.

11. Forgetting to Set the User ID Before a Commit

To save time, it is recommended that you set the user ID to the application manager's ID before you commit to ENVY using the Migration Assistance Tool.

If you set the user ID to the ID of the person who is to manage the application before you commit the application to ENVY, when the application is created, that person will automatically become the application manager and the class owner for all the classes within the application. This avoids having to go back later and change the application managers and class owners.

12. Forgetting to Save the Image Frequently

If you are running on a small machine, you might (most likely will) hang the machine several times while you are migrating. Be sure to do the following:

- Save the image frequently so that if your machine hangs, you do not lose a lot of work.
- Save the image before committing to ENVY.

13. Skipping the Pilot Change Cycles

Migration is not just the act of getting your source code into ENVY. You also need to determine how you will operate with this new library management system, for example:

- What your development process will be
- How you will record the code that has gone into production.

When you have done your pilot migration, use the migrated code to run through several pilot change cycles. For example, do the following:

- Make a set of assumptions about how your change process should work.
- Make some changes.
- Test whether your process works.

If it does not work, change your assumptions and try another pilot change cycle. Doing these pilot change cycles can be invaluable in gaining a better understanding of how ENVY works, as well as helping to identify tools that you might use to make your development processes easier.

Mistakes After Migration

14. Forgetting That a Pilot Really is a Pilot

A pilot really should be a pilot. You should plan on documenting what you have done. Especially for GUIs, you want to document any code changes you had to make so that you can make the same changes in other GUIs you migrate. In some cases, you might be able to make the changes in V2.2 so that you do not need to do anything when you actually do the migration.

In addition, you should plan to throw away the pilot and do the real migration based on what you have learned about naming conventions, new development processes, and so on.

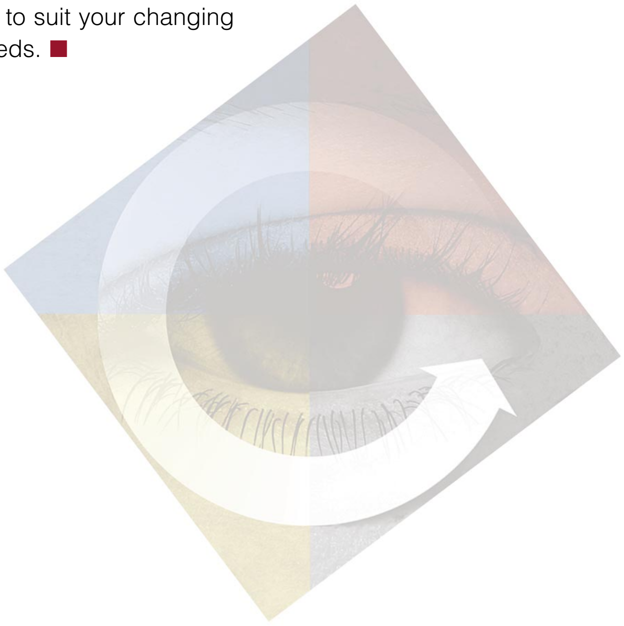
15. Forgetting Library Management Tools

Based on your pilot, you will likely want some tools to help manage your development process. These tools typically require Smalltalk skills, so plan to either acquire Smalltalk education or contract with a service provider for help in writing the tools.

You undoubtedly spent time developing procedures and tools for your MSL library system. You should expect to spend time putting procedures and tools in place for ENVY, too.

16. Locking into the Organization Chosen During Migration

Even after you perform your real migration, you are not locked into the application structure that you chose during migration. You can move parts from one ENVY application to another. There are just more rules to follow than when you moved parts from one application to another with the Migration Assistance Tool. As long as you follow the rules, you can change your application organization to suit your changing development needs. ■



Beyond Migration—Hints and Tips

by Jeri Petersen, VisualAge Generator Consulting Services and Nina Newton, VisualAge Generator Information Development

This article discusses some techniques that are useful in a development environment. Remember, these are hints and tips that we have found to be useful for VisualAge Generator as of V3.1 Fixpak 1.

Recommended Default Settings

The following shows the default settings that we have found to be useful when starting to work with your image. To set your default settings, do the following:

From the **VisualAge Organizer** window, do the following:

1. Select **Options,** then **Full Menus.** Full Menus enables you to see all the menu options.
2. Select **Options,** then **Preferences,** then do the following:
   a. On the **General page,** deselect the **Show Quick Start** window and select **Properties Table.**
   b. On the **VAGen-Map** page, set the **Supported devices.**
   c. On the **VAGen-SQL** page, set the **Database name** and **SQL date/time.**
   d. On the **VAGen-Test Linkage** page, set the **Linkage table part** and **Programs in library to bypass.**
   e. On the **VAGen-Generation** page, set **Generation options** and **Batch generation command**.
3. Select **VAGen Parts,** then **View,** then **Name and Edition.**
4. Optionally, select **Applications,** then **View,** then **Set Prefixes** and set the prefixes for commonly referenced applications. Then select **Applications,** then **View** and then **Matching Prefixes.**

The **Preferences** notebook enables you to set things such as your SQL database name, generation options part, and others.

Using **Name and Edition** for VAGen parts enables you see the time stamp for each VAGen part just below the part name.

For **Applications,** you might want to select **View,** then **Show VisualAge Applications,** or **View,** then **Show Applications Matching Prefixes.** This partly depends on how many ENVY applications you have and whether you typically work with all of the code or just one or two subsystems.

If you create a baseline image or if you have an empty image that each developer can copy to his or her own machine, you should set the user ID to unknown before you save the image. This keeps you from having to go to each developer's machine, enter your password and then stand by while the developer changes to his or her own user ID. We will explain how to do this later in this article.

Creating Your Own Workspace

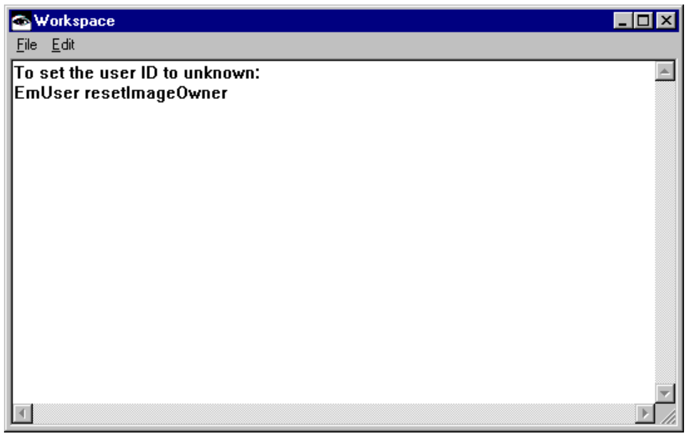
A *workspace* is a place in ENVY where you can save Smalltalk code that you want to be able to run again without actually making an ENVY application from the code.

To create and use your own workspace, do the following:

1. From the **VisualAge Organizer** window, select **File** and then **New Workspace.**
2. Enter the commands you want to remember.
3. Select **File,** then **Save As** and then specify a path and file name.
4. To go back to your work space later, from the **VisualAge Organizer** window, select **File,** then **Open Workspace** and point to the saved workspace.

Setting the User ID to Unknown

Figure 1 shows the first command you should store in your own workspace. This is the Smalltalk code to set the user ID to unknown. Type in this code, then swipe the code with your mouse, press mouse button 2 and then select **Execute.** This changes the user ID for the image to unknown, creating a default image that can be copied to each developer's machine.



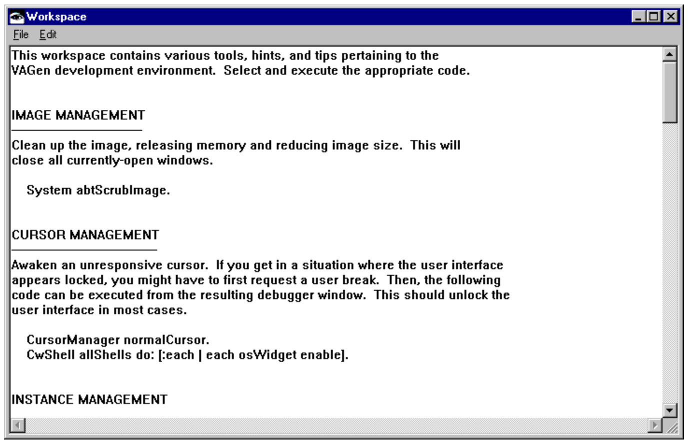
```
To set the user ID to unknown:
EmUser resetImageOwner
```

Figure 1

VAGen Workspace

VisualAge Generator provides a predefined workspace that you can access from the **System Transcript** window. This workspace (shown in figure 2) contains Smalltalk code that we think would be useful to you.

To access the VAGen Workspace, go to the **System Transcript** window, select **Tools** and then **Open VAGen Tools Workspace.**



```
This workspace contains various tools, hints, and tips pertaining to the
VAGen development environment.  Select and execute the appropriate code.

IMAGE MANAGEMENT

Clean up the image, releasing memory and reducing image size.  This will
close all currently-open windows.

    System abtScrubImage.

CURSOR MANAGEMENT

Awaken an unresponsive cursor.  If you get in a situation where the user interface
appears locked, you might have to first request a user break.  Then, the following
code can be executed from the resulting debugger window.  This should unlock the
user interface in most cases.

    CursorManager normalCursor.
    CwShell allShells do: [:each | each osWidget enable].

INSTANCE MANAGEMENT
```

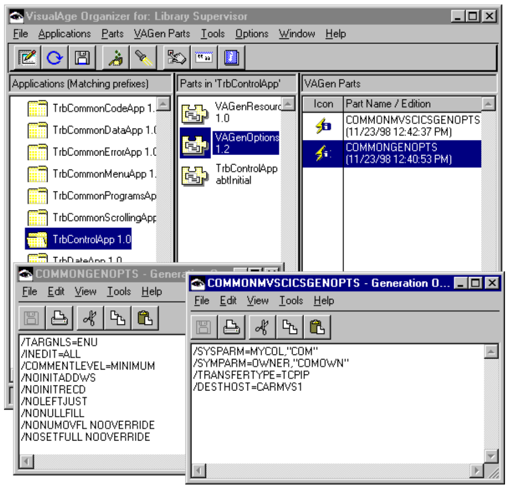
Figure 2

Developer Configuration Maps

Configuration maps are required for LAN generation. However, you do not necessarily want each developer to be modifying the system test level of configuration maps. This technique provides each developer with a private configuration map so that they can easily make changes.

To start, do the following:

1. Define your applications and your common generation options parts. In figure 3, there is a COMMONGENOPTS part that is the default generation options part, which is specified in the hpt.ini file. COMMONGENOPTS specifies generation options that apply to all target environments and indicates which generation options cannot be overridden. The COMMONMVSCICSGENOPTS part contains just those generation options that apply for the MVS CICS target environment.

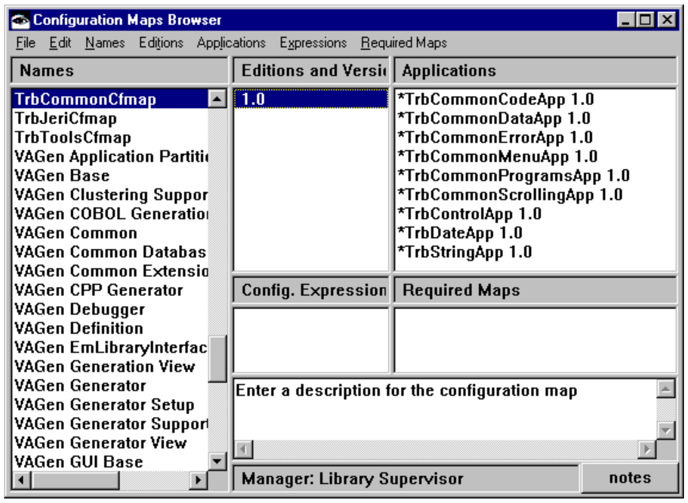


```
/TARGNLS=ENU
/INEDIT=ALL
/COMMENTLEVEL=MINIMUM
/NOINITADDWS
/NOINITRECD
/NOLEFTJUST
/NONULLFILL
/NONUMOVFL NOOVERRIDE
/NOSETFULL NOOVERRIDE
```

```
/SYSPARM=MYCOL,"COM"
/SYMPARM=OWNER,"COMOWN"
/TRANSFERTYPE=TCPIP
/DESTHOST=CARMVS1
```

Figure 3

2. Define a configuration map for your system. This configuration map contains all the code that is part of the system, including the application (TrbControlApp) that contains the generation options parts (as shown in figure 4).



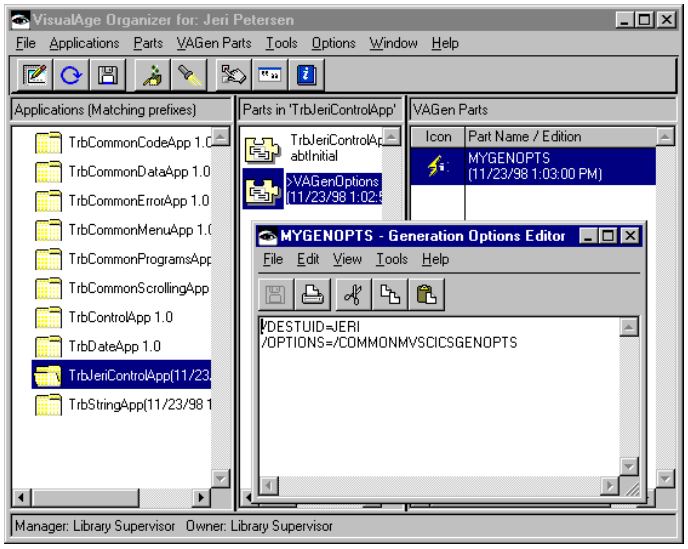
Names	Editions and Versi	Applications
TrbCommonCfmap	1.0	*TrbCommonCodeApp 1.0
TrbJeriCfmap		*TrbCommonDataApp 1.0
TrbToolsCfmap		*TrbCommonErrorApp 1.0
VAGen Application Partitic		*TrbCommonMenuApp 1.0
VAGen Base		*TrbCommonProgramsApp 1.0
VAGen Clustering Suppor		*TrbCommonScrollingApp 1.0
VAGen COBOL Generatio		*TrbControlApp 1.0
VAGen Common		*TrbDateApp 1.0
VAGen Common Databas		*TrbStringApp 1.0
VAGen Common Extensio		
VAGen CPP Generator		
VAGen Debugger		
VAGen Definition		
VAGen EmLibraryInterfac		
VAGen Generation View		
VAGen Generator		
VAGen Generator Setup		
VAGen Generator Support		
VAGen Generator View		
VAGen GUI Base		

Enter a description for the configuration map

Figure 4

3. Next, each developer defines his or her own control application to contain their private generation options. In this case, the application is called TrbJeriControlApp. One of the parts contained in this application is the generation options part that is unique to this developer. The developer's generation options part cascades to the generation options for the MVS CICS environment by using the /OPTIONS generation option.

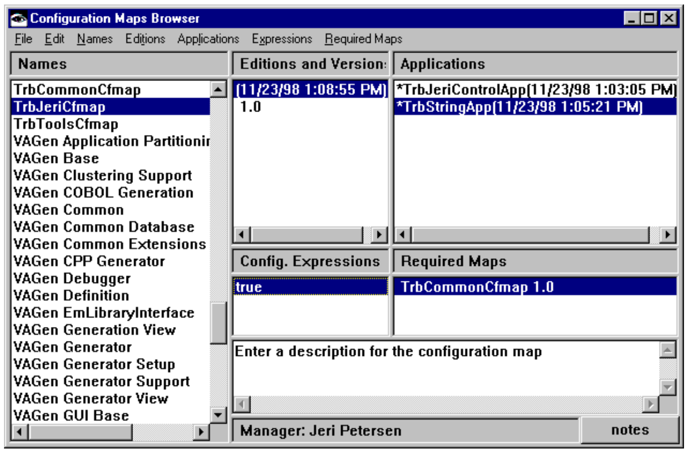
Now the developer is ready to make code changes. In figure 5, the changes have been made to TrbStringApp.



```
/DESTUID=JERI
/OPTIONS=/COMMONMVSCICSGENOPTS
```

Figure 5

4. The developer then creates a configuration map that includes his or her control application and the application that contains the code changes (see figure 6). The applications and the configuration map do not need to be versioned for LAN generation. However, be sure that the Windows NT Regional Settings are identical on all your clients, LAN generation servers, and your Library Manager server. Otherwise, the edition of the configuration map might not be found. The developer's configuration map specifies the normal test configuration map as a required map.


Configuration Maps Browser
File Edit Names Editions Applications Expressions Required Maps
Names
Editions and Version:
Applications
Config. Expressions
Required Maps
notes

TrbCommonCfmap
TrbJeriCfmap
TrbToolsCfmap
VAGen Application Partitionir
VAGen Base
VAGen Clustering Support
VAGen COBOL Generation
VAGen Common
VAGen Common Database
VAGen Common Extensions
VAGen CPP Generator
VAGen Debugger
VAGen Definition
VAGen EmLibraryInterface
VAGen Generation View
VAGen Generator
VAGen Generator Setup
VAGen Generator Support
VAGen Generator View
VAGen GUI Base

(11/23/98 1:08:55 PM)
1.0

*TrbJeriControlApp(11/23/98 1:03:05 PM)
*TrbStringApp(11/23/98 1:05:21 PM)

true

TrbCommonCfmap 1.0

Enter a description for the configuration map

Manager: Jeri Petersen

Figure 6

When the developer's configuration map is loaded with required maps, the normal test map and its applications are loaded first and then any applications in the developer's configuration map are reloaded for the version/edition specified by the developer's configuration map.

In practice, setting up the developer's control application and the developer's configuration map is a one-time activity. For most changes, the developer would just need to make the change and update his or her configuration map to contain the applications that they have changed.

Exporting ESF or LISTA

Many of you probably interface your MSLs to an existing library management system, most likely by exporting your esf at the time you generate. Because ENVY automatically saves editions of the VAGen parts whenever you make a change, you might not need to export esf any longer. Alternatively, just having a list associates might be enough. Other things to consider are:

- If you use LAN generation, you specify the configuration map name and edition on the GENERATE command.
- Generation loads the configuration map into the image on the LAN generation server and then does the generate.
- Unlike MSLs, you cannot just export the esf or do the list associates before the generate command. You need generation to run to get the correct configuration map loaded.

This technique shows how to split the generate and prepare steps and include an export or list associates in between the generate and prepare. It assumes that the /NOPREP generation option is never included in a generation options part, but only specified in the **Generation Options** notebook or on the command line.

EFKSERV is the REXX command that runs on the LAN generation server. The following shows some examples of code that are needed to implement this technique. You need to do the following:

- Check to see if /NOPREP was explicitly specified.
- If it was, you do not need to include it on the GENERATE command. The following is an example of the code:

```
/* Determine if /NOPREP was explicitly specified */
gen_command = gen_subcommand_string
preprc = POS('/NOPREP', gen_command, 1)
```

- If it was not explicitly specified, then you need to force /NOPREP so that you have the opportunity to do the esf export or list associates after generation, but before preparation occurs. The following is an example of the code:

```
/*  /NOPREP was not explicitly specified, force it /*
/*  on the GENERATE command */
if preprc = 0 then do
   symparm_value = symparm_value || '/NOPREP'
end
/*  Build Generate command back up */
if sub_comand< > 'PREPARE' then do /* add symparm for generate */
   symparm_value = "/SYMPARM=OUTPGEN," || ""'userid'""
   GEN_SUBCOMMAND_STRING=GEN_SUBCOMMAND_STRING  symparm_value
    end
```

- After the GENERATE command has completed, add logic to do the esf export or the list associates. The following example shows only the list associates. If you need to export esf, you first need to list associates and then use the output from the list as input to tell export what to do. There is no longer an export with associates.

```
prog_list = prog_name /* split prog_list into individual programs */
do while prog_list < > "
   parse value prog_list with program_name ',' prog_list
   lista_file = gen_listing_dir || program_name || '.LSA'
   lista_err = gen_listing_dir || program_name || '.LSE'
   lista_cmd = ,
     'LISTA /NAME=' || program_name || ' /TYPE=ALL /OUTFILE=' || ,
      lista_file
   'erase' lista_file '2>nul'  /* erase old .lsa  file */
   HPTCMD ' ' lista_cmd ' >>'lista_err ' 2>>&1'
   'erase' lista_err '2>nul'  /*no need to  save .lse file */
end                        /* end LISTA loop  */
```

- Finally, in EFKSERV, if /NOPREP was not explicitly requested, run the prepare command to send the outputs of generation for compilation. The code is as follows:

```
/* Run the prepare step if /NOPREP was not explicitly specified */
If  preprc = 0 then do
  prep_name = gen_listing_dir || prog_name || '.PRP'
   HPTCMD ' PREPARE ' prep_name  '>>'gen_listing_file '2>>&1'
end
```

The following example assumes that you are using an MVS host, but similar techniques would be possible for other target environments. You need to:

- Modify the template that controls the upload to MVS. The following code adds :TYPE commands to define transfer to the host:

```
:TYPE PART_TYPE='EZELSA ' WORKSTATION_EXT='.LSA'
   MVS_EXT='EZELSA'
```

  This code adds :CONTROL commands to cause transfer to occur:

```
:CONTROL
  HAS_DBCS='N'
  NAME='%ezegmbr%'
  TYPE='EZELSA'
  HAS_SQL='N'
  USE_EXT_ALT='N'
```

- On MVS, modify the CLIST that allocates the files for preparation outputs so the results of the esf export or list associates can be stored on the host.

Minimizing Display of Passwords

There has been a number of entries on the forums related to showing the passwords in the clear. This technique works if you are using LAN generation. However, be aware that this technique is release dependent. For this to work, you need to do the following:

- Assign a user ID and password to the LAN generation server machine.
- Create that user ID and password on MVS and set the password so that it never changes.
- Set up RACF to enable that user ID to update the developer's data sets where the outputs of generate will be placed.
- Modify EFKSERV to split generation and preparation.
- Change the PREPARE command to invoke EFKSNDxx directly, where xx is 30 or 31 depending on your version of VisualAge Generator. When you use EFKSNDxx directly, you can specify the /DESTUID and /DESTPASSWORD generation options for EFKSNDxx. This hardcodes the user ID and password in the REXX exec.
- Ensure that EFKSERV is on a drive and directory that the developers cannot see.

With this technique, the developers no longer specify /DESTUID and /DESTPASSWORD, because it is the user ID for the LAN generation server that does the upload to the host. The password does not appear in the .PRP file that is created by generation.

We hope you find the information in this article helpful in your development process. ■

Acronyms

3GL	third-generation language
4GL	fourth-generation language
AIX	Advanced Interactive Executive
API	Application Programming Interface
AS/400	Application System/400
CAE/2	Client Application Enabler/2
CASE	Computer-aided Software Engineering
CICS	Customer Information Control System
CICS OS2	Customer Information Control System Operating System/2
CPU	central processing unit
CSP	Cross System Product
DB2	Database 2
DBCS	double-byte character set
DBMS	database management system
DCE	distributed computing environment
DRDA	distributed relational database architecture
EMEA	Europe/Middle East/Africa
GUI	graphical user interface
IBM	International Business Machines
IMS	Information Management System
LAN	Local Area Network
MSL	member specifications library
MVS	Multiple Virtual Storage
NT	Notes
OS/2	Operating System/2
OS/390	Operating System/390
OS/400	Operating System/400
RAD	rapid application development
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
VM	Virtual Machine
VSE	Virtual Storage Extended
WWW	World Wide Web

Comment Form

Please check any appropriate boxes:

❏ I'd like to receive future issues of this newsletter.
(You need to check this item only if you have not already responded.)

❏ I'd like more information about Version 3.1.

❏ I'm interested in writing an article to include in *The VisualAge Generator Newsletter.*
Subject:

❏ I'm interested in participating in an AD users' group meeting.

❏ I'm interested in participating in a VisualAge Generator users' group meeting.

I have a question I'd like to submit for the Question & Answer section of this newsletter:

Are we putting the type of information you want to see in the newsletter?
If not, what would you like to see in the newsletter?

Any comments you'd like to share with us about VisualAge Generator or about this newsletter?
(Include your comments or concerns about VisualAge Generator's future directions here.)

Name	Title
Company Name	
Street Address/P.O. Box	
City	State/Province
ZIP/Postal Code	Country
Phone No.	FAX No.

Fold, tape, and mail this page - no postage is required. Or FAX it to (919) 254-0206.

G242-0315-10

IBM ®

Cut or Fold Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines
The VisualAge Generator Newsletter
Newsletter Editor
TF6B/062/J125
P.O. Box 12195
RTP, NC 27709-2195
USA

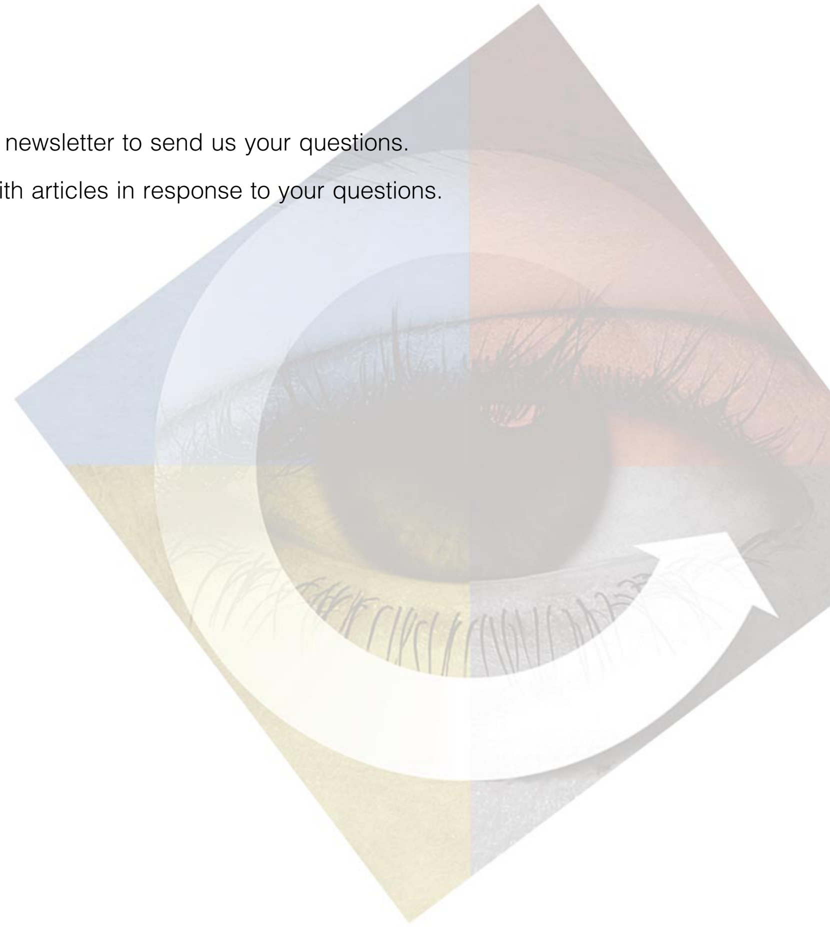
Fold and Tape | Please do not staple | Fold and Tape

Cut or Fold Along Line

G242-0315-10

A Question From Us To You

Do you have questions? If so, use the Comment Form in this newsletter to send us your questions. Then, in future issues of the newsletter we will provide you with articles in response to your questions.



The VisualAge Generator Newsletter

This newsletter is published by the IBM Software Solutions Division, Research Triangle Park Development Laboratory. Letters to the editor are welcome. Please address correspondence to:

The VisualAge Generator Newsletter
Managing Editor
IBM Corporation
Dept. TF6B/062
P.O. Box 12195
3039 Cornwallis Road
RTP, NC 27709-2195
USA
FAX: (919) 254-0206

The VisualAge Generator Newsletter
IBM Corporation
Dept. TF6B/062
P.O. Box 12195
3039 Cornwallis Road
RTP, NC 27709-2195
USA

G242-0315-10