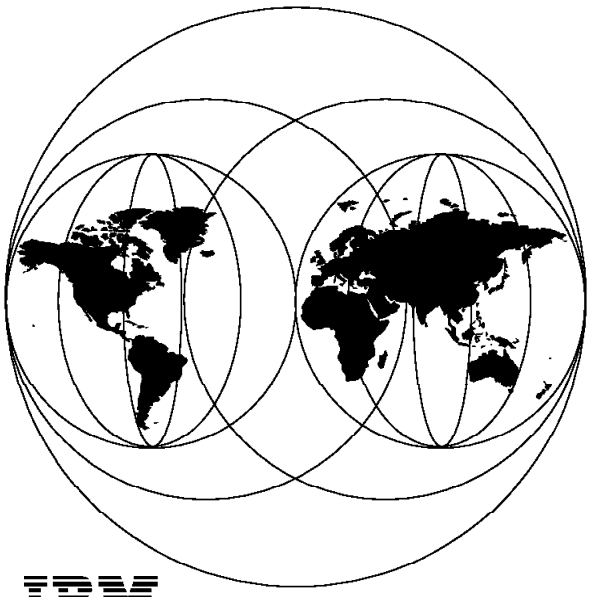
SG24-2219-00

VisualAge for Smalltalk Handbook
Volume 2: Features

September 1997



IBM
International Technical Support Organization
San Jose Center

SG24-2219-00

International Technical Support Organization

VisualAge for Smalltalk Handbook
Volume 2: Features

September 1997

Take Note!

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

First Edition (September 1997)

This edition applies to VisualAge for Smalltalk, Versions 2, 3, and 4, for use with OS/2, AIX, and Microsoft Windows 95/NT.

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

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

Contents

Preface

This book addresses many common questions in the VisualAge for Smalltalk development arena. It covers various aspects of VisualAge and IBM Smalltalk through answers to frequently asked questions, hints and tips from users and developers, and online bulletin boards inside and outside of IBM.

This redbook will help VisualAge for Smalltalk developers find answers to their everyday questions. The book provides usage guidelines for areas such as change management, performance, database access, and transaction processing, to help developers avoid common programming pitfalls.

The *VisualAge for Smalltalk Handbook* has two volumes:

- Volume 1: Fundamentals

  Volume 1 covers general programming questions on such topics as image maintenance, graphical user interfaces, naming conventions, and the IBM Smalltalk language.

- Volume 2: Features

  Volume 2 focuses on VisualAge for Smalltalk features, such as AS/400 Connection, Communications and Transactions, Distributed, Reports, and Web Connection.

This book is written for VisualAge for Smalltalk programmers, project leaders, and developers. Understanding many of the questions and answers in this book requires substantial knowledge of the VisualAge for Smalltalk product and the Smalltalk language.

This book does not discuss questions related to methodologies, analysis and design. For a discussion of object-oriented analysis and design with VisualAge for Smalltalk, refer to *Visual Modeling Technique*.

How This Redbook Is Organized

This redbook contains 348 pages. It is organized as follows:

- Volume 1: Fundamentals

  This book sets the stage for programming with VisualAge for Smalltalk. It contains questions and answers dealing with the base product and everyday programming pitfalls.

  - Chapter 1, “What Is VisualAge for Smalltalk”

    This chapter provides some non-technical discussion of programming with VisualAge for Smalltalk.

  - Chapter 2, “General Information”

    This chapter provides general programming tips, from image maintenance and packaging to troubleshooting.

  - Chapter 3, “Graphical User Interface”

    This chapter provides a discussion of graphical user interface parts, such as containers, notebooks, buttons, or tables.

  - Chapter 4, “IBM Smalltalk Programming Language”

    This chapter provides questions and answers regarding the IBM Smalltalk programming language.

  - Chapter 5, “ENVY”

    This chapter provides a discussion of the ENVY team programming environment.

  - Chapter 6, “Microsoft Windows”

    This chapter provides a discussion of programming issues related to the Microsoft Windows platform.

  - Chapter 7, “National Language Support”

    This chapter discusses national language support with VisualAge for Smalltalk.

- Volume 2: Features

  This book details on the various VisualAge for Smalltalk features, some of which are packaged with the base product and some others are separately orderable.

  - Chapter 1, “AS/400 Connection”

    This chapter provides a discussion of the AS/400 Connection feature and related items such as ClientAccess/400 and DB2/400.

- Chapter 2, “Communications and Transactions”

  This chapter provides a discussion of the Communications/ Transactions feature and its protocols, such as TCP/IP, APPC, CICS, or NetBIOS.

- Chapter 3, “Interface to External Routines”

  This chapter discusses the use of external routines, such as C or COBOL programs.

- Chapter 4, “CICS and IMS Connection”

  This chapter provides a discussion of the CICS/IMS Connection feature.

- Chapter 5, “Database”

  This chapter covers questions and answers related to database access from VisualAge for Smalltalk, such as DB2 or Oracle.

- Chapter 6, “Distributed”

  This chapter provides insight into the VisualAge for Smalltalk Distributed feature, for example, remote objects or object spaces.

- Chapter 7, “Reports”

  This chapter covers the VisualAge for Smalltalk Reports feature.

- Chapter 8, “SOM and DSOM”

  This chapter provides a discussion of the System Object Model and the Distributed System Object Model.

- Chapter 9, “Web Connection”

  This chapter discusses the Web Connection feature of VisualAge for Smalltalk.

ITSO on the Internet

Internet users may find additional material about new redbooks on the ITSO World Wide Web home page. Point your Web browser to the following URL:

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

IBM internal users may also download redbooks or scan through redbook abstracts. Point your Web browser to the internal IBM Redbooks home page:

http://w3.itso.ibm.com/redbooks

If you do not have World Wide Web access, you can obtain the list of all current redbooks through the Internet by anonymous FTP to:

```
ftp.almaden.ibm.com
cd /redbooks
get itsopub.txt
```

The FTP server, *ftp.almaden.ibm.com*, also stores the sample software from the diskettes accompanying some of the redbooks. To retrieve the sample files, issue the following commands from the */redbooks* directory:

```
cd SG24xxxx          ← Redbook number
binary
get SampleFile.EXE
ascii
get SampleFile.TXT
```

All users of ITSO publications are encouraged to provide feedback to improve quality over time. Send questions about and feedback on redbooks to:

```
redbook@vnet.ibm.com       or
REDBOOK at WTSCPOK         or
USIB5FWN at IBMMAIL
```

If you find an error in the software that is supplied with a redbook, please send a bug report with a description of the problem to

```
bugs@vnet.ibm.com
```

To receive regular updates on new redbooks and general IBM announcements, you can subscribe to the IBM Announcement Listserver. It automatically supplies an Internet e-mail user with timely new announcement information (titles and optionally the letter or abstract) from selected categories. To get started, send an e-mail to

```
announce@webster.ibmlink.ibm.com
```

The keyword SUBSCRIBE must be the only word in the body of the e-mail; leave the subject line blank. You will receive a category form and listserver details. To immediately start your subscription to, for example, AS/400 announcements, put the words SELECT HW120 in the body of the note. On the afternoon of an announcement day, you will receive e-mail with the announcement, along with a list of newly available redbook titles. To obtain a full abstract of a particular redbook, use GET SG242535 in the note.

VisualAge Support on CompuServe

VisualAge users are encouraged to use CompuServe® to ask questions about VisualAge and its features. When logged on to your CompuServe account, type GO VISUALAGE to get into the *IBM Workstation Rapid Application Development* (WRAD) conference. You will find sections to discuss the following and other VisualAge topics:

- Installation/Begin
- Communication/Languages
- Database
- AS/400 Connection
- Web Connection

About the Authors

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose, CA.

Andi Bitterer works as a consultant for the International Technical Support Organization at the Almaden Research Center in San Jose, California. He graduated with a degree in computer science from the Technical University in Darmstadt, Germany. Andi joined IBM in 1987 and worked as an application development specialist in large customer projects for IBM Integration Systems Services and at the German AS/400 Field Support Center. Since joining the ITSO in 1994, he has taught workshops worldwide on object-oriented application development and the Internet. Andi is the author of three other VisualAge for Smalltalk books published by Prentice Hall. You can reach him by e-mail at bit@acm.org.

Vincent Dijkstra works as an Education Consultant at the IBM International Education Center, La Hulpe, Belgium. He holds Bachelor of Arts and Master of Arts degrees in Technology and Policy from the Eindhoven Technical University. Vincent started working for IBM in 1992 on a learning resource to assist managers and teams in transforming their organizations. In 1994 he joined the IBM Object Technology University, where he teaches Smalltalk, VisualAge, and more recently Java and analysis and design techniques. You can reach Vincent by e-mail at vdijkstra@vnet.ibm.com

Boris Shingarov received his Master of Science degree in Quantum Fields at Moscow State University, M.V.Lomonosov, in Russia. He is the head of the Center for VisualAge and IBM Smalltalk, Dialogue/MSU, providing VisualAge consulting and services all over Russia. Currently, Boris teaches VisualAge for Smalltalk courses at MSU. He can be reached by email at boris@visualage.dialogue.msu.su.

Acknowledgments

This book has been a team effort par excellence, even though most contributors were not even aware of being a team. However, we want to acknowledge everyone, inside and outside of IBM, who, knowingly or unknowingly, added information to the collection of invaluable hints and tips in this book.

Thanks go to Dave Allison, Rainer Angstmann, Jean-Pierre Augias, Wayne Beaton, Jan Belik, Paul Berglund, Jonathan Bezuidenho, Debu Bhattacharya, Franco Biaggi, Peter Boonen, Chris Bosman-Clark, Toufic Boubez, David Bourke, Dave Bowman, Mamie Branch, Michel Brassard, Chuck Bridgham, Robert Brown, Jordi Buj, Jerry Callistein, Jon Capezzuto, Tony Carlier, Janelle Carroll, Ron Cherveny, George Chiu, Chris Clark, Eric Clayberg, Stephen Cooper, Ellis Covington, Tim Cowan, Mark Cresswell, Jim Crossgrove, Greg Curfman, James Curtis, Chris Davia, Kirk Davis, Gordon Davis, John DeBinder, Ricardo Devis, Bill Dickenson, Stef van Dijk, Sue Dubbeling, John Earle, Ralf Eberhardt, Kevin Egolf, Linda Fargo, Ahmed Fattah, Gerald Fischer, Hans Forsberg, Mike French, Annick Fron, Sabine Gaissert, Ken Gelsinger, Chris Gerken, Michel Giroux, Erick Godoy, Tom Gordon, Chris Grindstaff, Juergen Guenauer, Sven Guyet, Tim Hanis, Brad Hanks, John Hansen, Nicla Havrup, Chris Hayes, Frank Haynes, Pam Helyar, Andy Heys, Hal Hildebrand, Tim Hilgenberg, Andrew Hobbs, Steve Hobson, Bryan Hogan, Andreas Huber, Greg Hutchinson, Mike Jacobs, Charif Jaouhar, Chuck Jaynes, Christopher Joak, Greg Johnson, Tom Johnson, Jane Jones, Gary Karasiuk, Gijsbert Karens, Dan Kehn, John Kellington, Dave Kennedy, Brian Kent, Kelly Keplinger, Alan Knight, George Kober, Martin Kunz, Bruce Lambert, Micky Lim, Michael Linderman, Daniel Lipp, Grace Liu, Bill Lockard, Tom Logan, Mark Lorenz, Pablo Lorenzo, Sabina Luzzatti, Stewart MacLean, Bill Mathews, Cynthia McCrickard, David McGee, John McIntosh, Ted McKnight, Gerald Meazell, Eric Meredith, Jim Mickelson, Gonzalo Mourino, Pat Mueller, Gen Nagatsuka, Martin Nally, Binh Nguyen, Jimmy Nguyen, Achim Nogli, Ami Noyman, Patrick O′Donnell, John O′Keefe, Dan Ohlhaut, Omar Padilla, Jim Pendergast, Rosa Peral, Ralf Pfiszter, Enrico Piccinin, Oliver Picot, Guenter Pindhofer, Rene Plourde, Greg Plummer, Bob Poulton, Steve Reeves, Brian Remedios, James Rendell, Scott Rich, Cameron Roy, Jouko Ruuskanen, Jari Saari, Craig Setera, Dan Shaver, Ed Shirk, Allen Smith, Christine Smith, Craig Smith, David Smith, Larry Smith, Daniel Stainhauser, Jim Stewart, Jeff Stratford, Jean Talbott, Steve Tang, John Tobin, Mark Tompkin, Mike Toohey, Rick Trotter, David Twyerould, Timo Ullrich, Harold Wadler, Jim Wason, Christopher Webster, Tony Weddle, Ronny Weisz, David Whiteman, Mark Wilkes, Joe Winchester, Rory Woodward, Cindy Wotus, Glorious Wright, Alfred Wu, Barry Young, Ric Zapanta, Sherwood Zern, and Slavik Zorin, for their numerous contributions, questions or answers, hints and tips.

Many thanks go to Alexis Scott, Nancy Lewis, Paul Braun, and Shawn Walsh at the ITSO Raleigh Center, for their great administrative support, and to Pat Donleycott, ITSO Raleigh Center Manager, for hosting the residency.

Thanks also to Shirley Hentzell and Maggie Cutler for their outstanding editing of this book, to Liz Rice for her editorial assistance, and to Geoff Nicholls, without whose advice this book would not have a decent index.

Comments Welcome

Your comments are important to us!

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

- Fax the evaluation form found in "ITSO Redbook Evaluation" to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Home Pages at the following URLs:

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

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Chapter 1. AS/400 Connection

In this chapter, we discuss the VisualAge for Smalltalk AS/400 Connection feature.

Multiple Programs with a Single Remote Procedure Call

Question We wrote an RPG program that, upon receiving a control parameter, sends a program message to a message queue and waits for a response. The message queue is in break mode. An RPC part calls this program among others from Smalltalk several times with *invokeAsynchronous,* and before the message was answered, *resultIsReady* returned *true*.

Answer Do not change the program name on the RPC part. If you want to call another program, use another RPC part. If you are having problems with the *resultIsReady* returning *true* after the program name has been changed, it is because the *resultIsReady* attribute is queried from the AS/400 when the getter method is run by your application. It is not set when the program ends. This would require the AS/400 to somehow send a message to the PC, which it does not do.

RPC Part Sets Commit Boundary

Question I want to start an RPC part with an initial program that sets a commit boundary by issuing a Start Commitment Control (STRCMTCTL) command with the Change (*CHG) parameter.

Then I want to change the program name in the RPC part to run a series of update programs. The last program would do either a commit or rollback.

Answer Instead of changing the RPC part to run different update programs, why not have the initial program call the update programs? In other words, make the calling program start commitment control, do the transactions, and the commit or rollback. All of the work would be done on the AS/400 within the same job. I think the STRCMTCTL would take the form: STRCMTCTL LCKLVL(*CHG) CMTSCOPE(*JOB).

Alternatively, your RPC could wait on a data queue for parameters. This way you could pass transaction instructions to the asynchronous RPC and the AS/400 job would read them when ready. Completion could be indicated in another data queue in which the client waits on.

Connection Problem with V3R1

Question When I try to start a session from VisualAge for Smalltalk using one of the AS/400 features, I get an error and the debugger gives me:

> Unable to connect to remote host 5250PLU2 #APNOCONNECTIVITY - Error occurred verifying security parameters for 5250PLU2 #E4APPCINVALIDAPPCRC - APPC Error: Primary: CONV FAILURE FAILURE NO RETRY Secondary: OK

Answer The problem is with the configuration of the 5250PLU2 that VisualAge for Smalltalk for Smalltalk is attempting to use. From the transcript, try the following:

Smalltalk

```
AS400System reconfigureAllSystems
AS400System availableSystemNames inspect.

(This should return an ordered collection of all the systems you
have configured in CM/2.)

(AS400System named: 'sysname') signOn.
```

Also, make sure that you are actually signing on to the AS/400. Verify that

```
AS400System promptForUseridAndPassword
```

returns *true*.

AS/400 Communication Error

Question I want to start a communication from VisualAge for Smalltalk to the AS/400. The connection with PC Support works fine. Now I get the error: 'UndefinedObject does not understand,' I checked all the parameters of my record descriptions, I tested it with different examples, but I always get the same error if I want to access the AS/400.

Answer The key is the comma in the message. Your pool dictionaries might be corrupted. This could have happened if you attempted a packaging operation without copying the *.SWP files as stated in the packaging section of the AS/400 Guide.

Reinstall the AS/400 Connection feature. This should not take too long, but it will repopulate the pool dictionaries and fix the problem.

Strange Characters on Log-on Window

Question When I try to connect to an AS/400, I receive a log-on window with very strange characters and dimensions (I cannot see the words User ID or Password). I can still connect if I type a user ID and a password; also the ′LogOn was successful′ window has funny characters.

Answer This would happen if the client is missing the AS4RTE20.MRI file. The file is located in the run-time directory for the platform on the server. Copy the file to each client.

Quick Form from AS/400 Record Classes

Question I have retrieved a record description from the file LIB1/FILE1. VisualAge for Smalltalk creates a class LIB1FILE1. I use a part of LIB1FILE1 in a visual part. I use also the Quick Form function to visualize some of the fields of LIB1FILE1. Quick Form function generates one label and one text for each field, but the text generated does not have the length of the AS/400. Is there any way to rapidly build visual forms where entry fields have the same characteristics of the AS/400 fields they come from?

Answer We looked at that when we first developed the feature. Ideally, we would like to see decimal number attributes, string lengths, and so on be assigned to the entry fields with quick form. Unfortunately, it is not possible as the quick form operation uses the Smalltalk type (Decimal, String) and has no reference to the RECORD subclass. As a result, we cannot add this feature.

Communication

Question What part of CA/400 needs to be running for a VisualAge Windows application to run against the AS/400? On OS/2 I just need to have started CM/2 to have access, so it seems I will just need to have the router started.

Answer Correct. The minimum on Windows is the router. If the shared code library resides on shared folders, then folders will need to be started for the developers (not the application users).

Question Does VisualAge for Smalltalk work through the ASCII workstation controller (WSC) and CA/400?

Answer It should. As long as the router connection through the ASCII WSC supports APPC it should work. We use the router APIs so the router should make the physical connection transparent.

Read Next/Previous

Question I am using the AS/400 File Access Part and get what seems like strange behavior when using the Read Next and Read Previous methods. When I switch directions through the file using Read Next or Read Previous, the fields are still the same ones until I again invoke that method, which then retrieves the appropriate record. Is this the way it is supposed to work?

Answer This is working the way the AS/400 does (even though it is a bit confusing). The way you can think of file positioning is with a cursor. When the file is open, the cursor is positioned before the first record.

```
→          "Cursor is here after an open"
  Record 1
  Record 2
  Record ...
  Record x
```

When a *readNext* is done, the record directly below the cursor is returned to your program and the cursor is bumped:

```
  Record 1   "This record is returned by the readNext"
→          "Cursor position after the readNext."
  Record 2
  Record ...
  Record x
```

The next *readNext* would position the cursor like this after completion:

```
  Record 1
  Record 2   "This record is returned"
→            "Cursor is here after the readNext"
  Record ...
  Record x
```

In the above example, the last *readNext* returned Record 2. When a *readPrevious* is done, the record before the cursor is returned and the cursor is bumped. In this case Record 2 is returned again.

```
  Record 1
→            "Cursor is here after the readPrevious"
  Record 2     "This record is returned"
  Record ...
  Record x
```

SQL Statements

Question We use use Visualage on a Windows workstation. We want to execute SQL statements on the AS/400 database. We understand we need CAE. Does that mean that we need a server with DB2/2 to use DRDA? Or, is there a direct link between CAE and the database of the AS/400 ?

Our AS/400 is now on V2R2. With Client Access in V3R1, will we be able to use the ODBC driver included in Client Access?

Answer SQL access is available through the VisualAge for Smalltalk database parts, DB2/2 and a DDCS/2 gateway server. The AS/400 feature provides record level file access directly to the AS/400.

ClientAccess/400 Windows 3.1 is shipping with ODBC support. Through the support of ODBC by VisualAge for Smalltalk, you can also access the AS/400 with SQL directly through the ClientAccess/400 ODBC driver.

Note that you do not need the AS/400 Connection feature to access the AS/400 database through SQL.

Data Queues and Records

Question We have an application interfacing to an AS/400 that uses data queues. Using the FIFO example that comes with the VisualAge for Smalltalk AS/400 Connection feature we can write a record to the queue. However, we cannot do the reverse, which is to read from the queue (dequeue) and put the result into a record. In that case, what comes back from the AS/400 is not the record that we expect (it could be an error message), is there a way to interrogate the dequeued object before assigning it to a record?

Answer The data queue parts enqueue records but dequeue AS400DataQueueEntry instances. You can see this in the parts reference (FIFO Data Queue...Actions...#dequeue Result type). Since the AS400DataQueueEntry class is not in the parts reference, you need to do a little digging.

Ideally, enqueue and dequeue should have data symmetry. That is to say, it would be nice if, when you enqueue a record, you could dequeue a record. Unfortunately, this is not the case. The dequeue operation can include sender information. Since this information is not part of the enqueue operation, we lose symmetry.

When a dequeue action is performed, the #result attribute is the dequeued data. This dequeued data is an instance of AS400DataQueueEntry. The AS400DataQueueEntry has six attributes: data, jobName, jobNumber, key, userName, and userProfile. The bulk of these attributes are associated with the sender information (the job which enqueued the data). The #data attribute will contain the record that you are looking for.

The FIFO example is a bit misleading because when you dequeue the order data, the result is added to the OrderedCollection and the list box appears to show a record. In reality, instances of AS400DataQueueEntry are being added to the OrderedCollection. If you open the settings of the list box you will find that the attribute name is set to ′data.′ This will effectively show the record in the list box.

ODBC Requirements

Question When I want to use the ODBC support of VisualAge for Smalltalk, do I still need DDCS/2 and DB2/2 on our clients to be able to access the AS/400 (level 3.1) data with SQL? What would be the advantage of using DDCS/2 and DB2/2, if any?

Answer DDCS/2 and DB2/2 are needed only if you are not using ODBC and you want to access different types of databases at the same time. If your data base is of one type (only AS/400 systems) then ODBC is the way to go. ClientAccess/400 V3R1 supports an ODBC driver for the AS/400. ODBC support is the best way to access AS/400 data using SQL.

ClientAccess/400 Optimized for OS/2

Question We are developing VisualAge for Smalltalk applications with AS/400 access parts and have two questions about CA/400 and the clients who will use the developed application. There are two OS/2 features of CA/400: the old 16-bit client and the new 32-bit Optimized for OS/2 client. My questions are these:

- Is CA/400 all that the clients will need to successfully execute the VisualAge for Smalltalk applications? That is, no need to purchase CM/2 because all of the required pieces of CM/2 will be included with CA/400 so that the VisualAge for Smalltalk application can successfully run?
- If so, are there any differences between the two CA/400 OS/2 clients that could cause difficulties in running the VisualAge for Smalltalk application.

Answer You don't even need to have CA/400 installed to use VisualAge for Smalltalk with the AS/400. CM/400 comes with the new OS/2 client so you shouldn't need to purchase separate CM/2. The AS/400 feature only needs the APPC components of the CM products. Since OS/2 optimized CA/400 includes this function, you do not have to purchase anything else.

VisualAge Server on LAN Server/400

Question Can I install the VisualAge for Smalltalk server on a LAN Server/400 now available, or do I have to install also the OS/2 LAN Server?

Answer VisualAge for Smalltalk is running fine on LAN Server/400, because inside LAN Server/400 it is actually OS/2 LAN Server.

Native Application Accessing VisualAge for Smalltalk Application

Question In our scenario, we want to have a user running a native AS/400 application (green screen). When the user presses a certain function key, it would cause a VisualAge for Smalltalk developed application to get focus and do something. The native AS/400 application would need to pass data to the VisualAge for Smalltalk developed PC application. We are thinking about using data queues to pass the data. However, how do we notify the VisualAge for Smalltalk application that something happened (in this case, that a new entry was added to the data queue), and cause the VisualAge for Smalltalk application to get focus?

Answer The VisualAge for Smalltalk application would have to be started already and waiting on the data queue. You can set the wait time so the VisualAge for Smalltalk application will block (that is, not run any further) until data is added to the data queue.

Packaging Problem

Question I have problems if I want to package an application with using the AS/400 classes from VisualAge for Smalltalk AS/400 Feature. If I run the packaging, I get an error. After that, if I start to work with my application in the development environment with access to the AS/400 and with the help of the AS/400 classes I always get an error that I cannot connect to the AS/400. If I shut down VisualAge for Smalltalk and I start it with the old image before I call the packaging, I get the connection to the AS/400. What is the reason for this error?

Answer If you packaged a run-time image and you did not copy the E4*.SWP files to the same directory as your development image, as called for in the *AS/400 Guide* (see packaging considerations, Run-time image packaging, Required run-time image files) you will get a walkback. This walkback will be something like *CfsError does not understand lseek:whence:...*

You will no longer be able to connect to the AS/400 in the development environment until you reinstall the VisualAge for Smalltalk AS/400 Connection feature into your image. After the install has completed, save your image. You should be able to connect in the development image now.

If you are getting a walkback in a packaged image that is something like *UndefinedObject does not understand,* (note the comma) then you need to copy the E4*.SWP files to the same directory as the run-time image. This is

documented in the *AS/400 Guide* (see Packaging considerations, Run-time distribution, Required run-time files).

Run-time Prerequisites for VisualAge for Smalltalk AS/400 Application

Question What are the prerequisites to run applications developed with VisualAge for Smalltalk and accessing the AS/400 (any AS/400 objects) in OS/2 and Windows environments. Is only Client Access supported or is other software also required?

Answer If your AS/400 runs on OS/400 V3R1, you only need to have an APPC LU 6.2 connection from the workstation to the AS/400. If you run OS/2, you can use CM/2 to establish the connection. Under Windows, use the ClientAccess/400 router. If you are using OS/400 V2R3 or OS/400 V3R0M5 you will also need the VisualAge for Smalltalk Server for OS/400 PRPQ (5799-FNN).

Fixes Available Via FTP

Tip The latest updates for VisualAge for Smalltalk are available through anonymous FTP and CompuServe. We recommend that users FTP the package to avoid line charges for large files. Please be sure to get the file readme.tm. It contains the instructions for installing the fixes.

The fixpacks are cumulative, so they contain all fixes from previous fixpacks.

Important!

You must apply the base fixes before applying any other fixpack.

To FTP the package, use the following commands:

Enter `ftp ps.boulder.ibm.com`
Enter `anonymous` as a userid.
Enter your email address for a password.
Enter `cd ps/products/visualage/fixes`
Change to the directory of your VisualAge version.
Enter `dir` to display all available files.
Enter `bin`
Enter `prompt n`
Enter `mget *` to get all of the fixpacks, or use
`get <fixPackName>` to get a particular fixpack.
Enter `quit` to exit FTP.

Multiple AS/400 Connections

Question We want to use Rumba for connectivity to the AS/400 because the five-session limitation of PC Support is too restrictive. I assume that VisualAge for Smalltalk would have the same restriction, as it relies on PCS. Will VisualAge for Smalltalk work with other connectivity options (like Rumba) to allow customers to connect to more than five systems within an application (at the same time)?

Answer The five-session limit is part of Work Station Function (WSF) and not a limit in the router. Rumba simply supports more terminal emulation sessions than WSF. This has nothing to do with the router limit.

The VisualAge for Smalltalk AS/400 Connection feature uses the router and not WSF, so we are bound by the limits of the router. We have tested and support only the PCS and CA/400 routers, although customers are successfully using other routers.

We have users running OS/400 V2R3 and V3R1 to attach to seven or eight AS/400 systems at the same time.

For a twinaxial link, the limit is one actual connection; however, through the use of an ADRS entry you can use that one connected system to route you to other systems. The system you are connected to must be a network node (*NETNODE). You can check this by doing a Display Network Attributes (DSPNETA) command. If it is a *NETNODE, then this will allow that AS/400 to do routing for you.

As far as a token-ring or Ethernet connection, the most TRLI entries that you can make is six. Again, however, you can take advantage of the ADRS entries to get more connections by routing through the systems that you are connected to.

Refer to the *PC Support/400 DOS Installation and Adminstration Guide* (SC41-0006) for further details on how to code the TRLI and ADRS entries.

Accessing AS/400 Physical File Members

Question For our existing AS/400 database we want to create a new VisualAge for Smalltalk front end. The database design includes heavy use of multiple members per physical file. How do I access those members through VisualAge for Smalltalk?

Answer Accessing data within a database member is possible with a bit of code. The manual does not indicate that this can be done. Instead of setting the file name within the file part's settings view, a *filename(mbrname)* should be set using this script:

```
Smalltalk
aFilePart fileName: 'ORDERS(JANUARY)'.
```

This will work when accessing data in existing members.

Accessing AS/400 Database

Question When accessing the AS/400 database I can use either SQL or read and write statements. Can I go either way with the VisualAge for Smalltalk parts?

If we use read and write, is this done using DDM, and how will it be checked that the user of the VisualAge for Smalltalk application is authorized to do this?

Answer The AS/400 feature provides record-level read and write access to the AS/400 data base. DDM is used to provide this support.

When the VisualAge for Smalltalk AS/400 feature needs to connect to the AS/400, a sign-on dialog is shown where the user must enter the user ID and password. This information is used to start DDM. Any subsequent access to the data base is based upon the authority of the user profile information provided at this time.

SQL access is available through the database features of base VisualAge. I recommend using the ODBC support of VisualAge for Smalltalk and ClientAccess/400 to access the AS/400 data base with SQL.

QENVY/QENVAUXD

Question When submitting a job via *invokeAsynchronous* under VisualAge for Smalltalk to an AS/400 under V3R0M5, QENVY/QENVAUXD is called with three parameters of QENVY, 000000, and 001350. The submitted job is using the AS/400 job description (JOBD) contained in the user ID of the AS/400 user. Where do we find more details on QENVAUXD or its equivalent for Version 3 R1 Host Services to control which job description and batch job queue is used for execution on the AS/400? Note that we are attempting to control where the invoked asynchronous job executes and do ***not*** want to use the user profile's job description.

Answer You can control the job description used for *invokeAsynchronous*. Before invoking the Remote Procedure Call (RPC), set the *jobDescription* like this:

Smalltalk

```
| rpc |
rpc := AS400RemoteProcedureCall new.
rpc programName: 'YOURPGM'.
rpc agent jobDescription: 'NEWJOBD'. "NEWJOBD job desc must exist"
rpc invokeAsynchronous.
```

Code Page Translation

Question We are running a VisualAge for Smalltalk application that accesses an AS/400 database in the Thai language. However, we found that the data is not translated correctly. We suspect a code-page problem. Our AS/400 has codepage 874-838 and my workstation has 437. How do we set up a correct translation?

Answer In order to display Thai data on a workstation, the workstation must be configured to display the character set of the AS/400 data. Using a code page of 437 (English) with an AS/400 codepage of 838 (Thai) is not a compatible combination. The character set included in 437 does not contain Thai characters. The 874 codepage does not appear to be an AS/400 code page, but a workstation code page.

According to *National Language Design Guide Volume 2*, SE09-8002, the recommended combinations for Thailand are the following:

AS/400 - Character set 1176, codepage 838, CCSID 838

PC (Primary) - Character set 1176, codepage 874, CCSID 874
(Secondary) - Character set 966, codepage 874, CCSID 4970

You will need to configure the workstation to use 874, using either the 1176 or the 966 character set.

The VisualAge for Smalltalk AS/400 Feature uses the local code page and the host job code page to create conversion tables. The conversion tables are built on the AS/400 and transferred to files on your workstation. The conversion tables are unique to the local code page and host code page combination.

Date Conversion

Question Can anyone give me any suggestions for the way to convert date into zoned decimal six-digit format? Any methods in Smalltalk we should look at?

Answer Yes. Try this as an example:

Smalltalk

```
| aDate day month year dayString monthString yearString decimalDate |
  aDate := Date today.
  day := aDate dayOfMonth.
  dayString := (day printString size = 1)
       ifTrue:['0',day printString]
       ifFalse:[day printString].

  month := aDate monthIndex.
  monthString := (month printString size = 1)
        ifTrue:['0',month printString]
        ifFalse:[month printString].

  year := aDate year.
  yearString := year printString copyFrom: 3 to: 4.
  decimalDate := (dayString, monthString, yearString) asDecimal
```

Database Transactions

Question My *KeyedFileView* opens a physical file as read and write and the lock mode is shared with readers and updaters. If I run *KeyedFileView new openWidget* twice to start two transactions to open the physical file, they cannot access the same record (record lock). However, readNext works correctly for each of the windows (two file pointers?).

My questions are these:

1. How many DDM server jobs are used by the VisualAge for Smalltalk to process these two transactions (windows)?
2. Is there a way that I can start a separate job for each transaction within one application?
3. Can I start each of the database transactions (not external functions) within an application on a separate process?

Answer

1. One DDM job is started for *all* file parts within an image.
2. You cannot start a separate job by using DDM. If you really need transaction support, you should use the RPC part and an AS/400 program to complete the transaction.
3. Yes, provided your application uses semaphores to synchronize the processes. This is necessary because the DDM job assumes a single conversation.

ReadAll Method

Question I wonder why, in the compound key example in the AS400Examples, you chose to write a script instead of using readAll. Is it because performance penalties are incurred when we use readAll?

Answer To the contrary, readAll is very fast. The reason it was not used in the example is that the example required the file to be open at the same time. ReadAll will not work if the file is already open.

Blocking Factor

Question The *VisualAge for Smalltalk AS/400 Connection feature* manual says that a blocking factor is used only when the files are opened as read only. Is that true?

Answer That is correct. OpenReadWrite sets the blocking factor to one for data integrity reasons. OpenReadOnly calculates a blocking factor based upon the size of the record description of the file. A read operation must make a request to the AS/400 and receive a reply back. The size of the reply must be contained within one APPC packet (about 32 KB). Because data translation is usually the performance bottleneck and not the actual data transfer from the AS/400, we constrained the calculated reply size to 2 KB. The calculated blocking factor will be approximately 2 KB divided by the size of the record description.

In practice, you should not change the blocking factor unless you have observed a performance problem with the calculated size.

Multitasking with the AS/400 Parts

Question I have a big file from which I have to retrieve all its records. Because this is a lengthy process, I decided to fork the reading of the records. The application's user interface is separated from the logic and file handling.

When I'm doing the read operation in a forked process, I can't write to the file from another process. I also can't read another file at the same time. I tried the following:

```
Smalltalk
openMyFile
"open the AS/400 file"

 semaphore := Semaphore new.      "semaphore is instance variable"
 semaphore signal.
 self signalEvent: #openTheFile.  "in the Composition Editor it
                                   goes to the openReadWrite action
                                   of the AS/400 file. "
 readFileContents
 " reading the file records "

 [ semaphore block.
 " the loop of reading the file records
   (also signaling for activities done
   in the Composition Editor)

 semaphore signal ] fork.
```

When I run this piece of code, the result is that I am blocked. Why? I need to handle few files from few parts at the same times. Should I be using the same semaphore?

Answer

Instead of using *Semaphore new*, use *Semaphore forMutualExclusion.* Then use *critical:* where you want the code to have exclusive access to the DDM conversation. For example,

```
Smalltalk
| semaphore |
semaphore := Semaphore forMutualExclusion.
[semaphore
 critical:[ < access DDM here > ]] fork.
```

Each critical bit should have its own semaphore, and the amount of work done within the critical block should be kept to a minimum.

Refer to *IBM Smalltalk, The Language* by David Smith. The book contains extensive coverage of this topic.

Checking PCS Status

Question I noticed that many of the errors I got when I run VisualAge for Smalltalk with AS/400 parts are coming from PC Support (PCS), for example, PCS is not started or PCS is not in good condition.

How can I check in a VisualAge for Smalltalk program whether PCS has been started and if it has, whether it is working okay before I use any AS/400 functions?

Answer The expression *E4Service remoteConnections* returns a collection of the names of the systems you can connect to. It uses the EHNAPPC_QuerySystems API in the ClientAccess router DLL to get the information.

Suppressing AS/400 Sign-on Dialog

Question In our application, we do not want the AS/400 sign-on screen to be shown, because we developed our own sign-on screen. I am trying to log on to the AS/400 after I opened the widget of the main program. I then provide the user ID and the password in the script. How can we suppress AS/400 sign-on dialog?

Answer According to the AS/400 System entry in the Parts Refererence of the VisualAge for Smalltalk AS/400 Feature User′s Guide, AS400System has an attribute called *promptForUserIdAndPassword*. This is a boolean attribute. If the attribute′s value is *true*, users will be prompted for their user ID and password the first time that VisualAge for Smalltalk attempts to access the AS/400. If the value is *false*, users will not be prompted for their user ID and password at that time. Instead, VisualAge for Smalltalk queries Communications Manager/2, ClientAccess/400, or PC Support/400 as appropriate for the user ID and password.

Changing Normal Cursor to Busy Cursor

Question I′d like to change the normal cursor to the busy cursor whenever I use AS/400 functions, because it is sometimes difficult for the end user to know whether the operation is finished or not. How should I do this?

Answer You should use the following construct when executing your long-running AS/400 access methods.

Smalltalk

```
System abtShowBusyCursorWhile:
    [ stuff that takes a long time to do ]
```

Do not use the *showBusyCursorWhile:* method, because it will not be packaged in your run-time image. VisualAge for Smalltalk uses this method only at development time.

PromptForUserIdAndPassword

Question I am using VisualAge for Smalltalk Team, OS/2 Warp, CM/2 and all the fixes applied (base fixes, warp fixes, AS/400 fixes). When I start CM/2, I have to provide a user ID and password to connect to the AS/400. However, I would like VisualAge for Smalltalk to query CM/2 for its user ID and password and use the same information to connect to the AS/400. Therefore I set *promptForUserIdAndPassword* to *true* in the AS400System class.

Unfortunately, this seems not to work, since I receive a VisualAge for Smalltalk debugger signalling a security error when starting the conversation. Any hints?

Answer First of all, you need to set *promptForUserIdAndPassword* to *false* so that VisualAge will query CM/2 for the sign-on information. You also need to make another change to your CM/2 configuration, assuming you are running CM/2 1.11 (not sure if this applies to CM/2 1.1 or 1.0):

1. Double-click on **Communications Manager Setup** from the CM/2 folder.
2. Click on the **Setup** button from the CM/2 Setup screen.
3. Select your configuration file name and click on **OK**.
4. You will then be presented with the next panel where there is an **Options** pull-down menu. Select that menu.
5. From the **Options** menu select **Configure any Profile or Feature**.
6. From the next panel scroll down to **SNA Features** and select it, then click on **Configure**.
7. On the next panel select **Conversation security** and then click on **Create**.
8. Click on the **Utilize User Profile Management** selection button and click on **Add** and then **OK**.

9. Close CM/2 Setup and let it run Verify.

If you had CM/2 running while you made this change, you can just select to dynamically update your connections. However, I would recommend stopping CM/2 and ending VisualAge for Smalltalk, then restarting them to ensure all is initialized correctly.

Performance Comparison

Question We have some concerns about the comparative performance of VisualAge for Smalltalk versus legacy RPG code. We have done a test whereby 300 objects were instantiated from an AS/400 database file. The time required to perform this operation through an inspection process was approximately 23 seconds. As a comparison, an RPG program that read the same 300 records into an array took only one second, making this quite a significant performance difference. Does that sound like what you know and expect?

Answer Here are a few things to consider about measuring performance.

- Use *Time millisecondsToRun: aBlock* for your measuring.
- Using inspectors will add several seconds to your measurement.
- For optimal performance, use *openReadOnly* instead of *openReadWrite*. The reason is that *openReadOnly* will access blocks of records, where *openReadWrite* will access records one at a time. The key point is to reduce the number of network turn-arounds. Also, see the discussion on *blockingFactor*, below.
- VisualAge for Smalltalk will ***never*** be as fast as RPG because the data has to come across the network and be translated to Smalltalk objects. For this reason, you must minimize the number of network turn-arounds.

The file parts maintain a cache of records to improve performance. The number of records maintained in the cache depends on the way the file is opened. If the file is opened for update, then the cache is set to one for data integrity reasons. If the file is opened with *openReadOnly,* then the *blockingFactor* attribute determines the cache size.

For openReadOnly, the file parts attempt to anticipate the need for data by accessing blocks of records. The number of records in the block is based on the size of the cache. The size of the cache can affect performance in two different ways. First, the number of network exchanges can be reduced by increasing the cache size. But as the size of the cache rises, application performance can decline. This is because the records are translated to Smalltalk objects as the data is received from the AS/400. If too many

records are accessed, but never used by the application, workstation performance can decline.

If your application does not set the *blockingFactor* attribute, a default value is calculated. The default value calculation is the integer result of dividing 2048 by the byte length of the *recordDescription* plus 16. If the byte length of the *recordDescription* attribute plus 16 exceeds 2048, the cache is set to one.

If the record length is less than 2032, consider setting the *blockingFactor* attribute to a value less than the default. If the record length is greater than or equal to 2032, consider setting the *blockingFactor* attribute to a value higher than the default. In any case, the determination of the best value is based on the resulting application performance.

For openReadWrite, because of the small cache size for files opened with *openReadWrite,* performance can suffer if a great many read actions are run. This is because each read consists of a network exchange with the AS/400. Consider using two file parts:

1. One file part for update operations. Read the record of interest before updating to ensure that proper locks are placed on the data. Move the cursor to the beginning or end of the file after the update is done to release the record lock.
2. A second file part is devoted to read operations. Do the majority of file read operations with this part.

As for measuring performance, here's an example:

Smalltalk

```
| myFile results records |

myFile := AS400KeyedFile new.
myFile
  fileName: 'CUSTMAST';
  libraryName: 'PROD';
  recordDescriptionName: #ExampleCustomerMasterRecord.

records := OrderedCollection new.
myFile openReadOnly.
results := Time millisecondsToRun:[
  1 to: 300 do:[records add: (myFile readNext)]].
results inspect.
```

Packaged Application Requirements

Question If we package an application that accesses the AS/400, does the PC that the application is installed on require OS/2 Communications Manager software installed in order for the package to run?

Answer Yes. VisualAge for Smalltalk provides the application and presentation layers, but the transport layer of communication and below must be provided by Communications Manager (on OS/2 platform) or Client Access (on Windows platform).

VisualAge for Smalltalk with OS/2 Client

Question Can Client Access/400 for OS/2 Optimized Client be used for the client transport services in the client workstation executing the deployed VisualAge for Smalltalk application? Is it correct that the only time you really must have CM/2 instead of Client Access/400 for OS/2 Optimized Client is for:

- 5250 Terminal Emulation (WSF type function), assuming no other 5250 emulator is installed?
- 3270 Terminal Emulation for connect to other server platforms?

Answer Only the OS/2 router is needed for any of the VisualAge for Smalltalk AS/400 services except EHLLAPI (terminal emulation/screen scraping) services. The router function is provided by the less-expensive Communications Manager/400 product, which does come bundled with Client Access/400 for OS/2 Optimized Client. Both CM/2 and CA/400 for OS/2 have EHLLAPI functionality.

So, for a deployed VisualAge for Smalltalk application, these are the options for OS/2 transport services:

VisualAge Client Application	Minimum Required Communication Software
Without EHLLAPI	Communications Manager/400
With EHLLAPI	Client Access/400 for OS/2
	Communications Manager/2

Packaged Image Size

Question My customer has developed a relatively simple VisualAge application (20 classes, 5 of them visual). Using the packager (as run-time image), the packaging process takes 1 hour 20 minutes on a PC with 75 MHz Pentium and 24 MB RAM, and the resulting image (with the unnecessary classes removed) is 8.5 MB. The applications that I've developed are much larger (accessing DB2 directly); they take 10 minutes to package and the image is around 2.5 MB.

Is this all because of the AS/400 feature? Any hints on reducing the image size?

Answer The AS/400 Feature classes will add some bulk (and therefore packaging time) to the packaged image size, but 8.5 MB seems quite large for such a simple application. There are a number of things that you can look for:

1. One of the first things to check whenever you have packaging problems is that the application's prerequisites are correct. The packager uses the prerequisite chain in searching for classes and methods that are required by the packaged image. If prerequisites are missing, required classes and methods will not be included. If applications are included in the prerequisites that are not actually required, there is a chance for extra classes and methods to be included. Check especially for prerequisites including the edit applications. You should have the run application as your only prerequisite.
2. Check for class variables and globals that may be initialized. When possible, such variables should be lazy-initialized at run-time so that they are not dumped into the packaged image. These variables can be automatically set to nil during packaging by placing an implementor of *prePackagingActionsFor:* on the application's class.
3. Finally, check the *.ES files that are written during the packaging process. These files provide statistics and information about exactly what was written to the packaged image. Should you find a class included that you don't believe should be there, then you need to find out what is referencing that class.

Transaction Program

Question We are using the AS/400 Connection feature under Windows. Our AS/400 runs under OS/400 V3R1. We can't get a connection to the AS/400. Is there a transaction program we must first activate?

Answer Check to see if an object of type *PGM with the name QEVYMAIN exists in library QIWS. If there is, then the transaction program is installed. If not, then you have to install the OS/400 Host Servers feature of OS/400 (5763SS1).

AS/400 Connection Feature Installation

Question When I install the AS/400 feature for Windows, I get the message "Warning 86 waiting lock, do not interrupt" and then the system seems to stall. What is going on?

Answer There are two likely causes:

1. Some VisualAge for Smalltalk Team user (not necessarily the client PC performing the installation of the AS/400 Feature) is accessing the library file in single-user mode. This may be the case if someone has just finished installing the library updates for the AS/400 feature and has not released the lock on the file yet (assuming no one but the installer should be connected to the library file during the installation). Or, if that is exactly what you, as the installer, are trying to do when you get this message, be sure no other users are currently connected before installation.

   If you get this message during the installation of the AS/400 feature on a client PC, then check your ABT.CNF start-up script in the VisualAge for Smalltalk directory (usually VISUALAG) to make sure the attributes of your EmLibrary are set as follows:

```
Smalltalk
"Start up Library."

 EmLibrary
   defaultName: " ...Your library's name should be here." ;
   releaseLockMode: true;
   singleUserMode: false.
```

If *singleUserMode* and *releaseLockMode* are correctly set, then it may be that one of the other library users has not correctly set these in their ABT.CNF files to allow shared use. Check with your VisualAge Team server administrator.

2. To share files in Windows, you must have SHARE.EXE loaded before your network log-in and Windows start-up. Check your AUTOEXEC.BAT file in the root directory of your boot partition. It should have SHARE.EXE as the first executed program. If it is not in AUTOEXEC.BAT, it may be executed in an ″INSTALL=″ statement in the CONFIG.SYS file, so check that file also. See your DOS Command Reference for usage details.

   If SHARE is present but you still have this problem, SHARE may have run out of resources. Try enlarging the /l or /f values. Also, be sure that all library file users have SHARE loaded. A quick way to check is to execute ″mem /c″ at a DOS prompt.

ILE Service Programs

Question Is it possible for VisualAge for Smalltalk to call an ILE service program function, that is, an entry point defined in an ILE program?

Answer You cannot do this directly. You would have to write a wrapper ILE program that calls the needed functions. Then it is a straightforward remote procedure call (RPC) to the default entry point of the wrapper program.

This is not a limitation of VisualAge for Smalltalk or the AS/400 Feature. Functions in ILE service programs can only be called by ILE programs that are bound to the service program during the create program process. Although the (OS/400 V3R1) RPC server program used by the VisualAge for Smalltalk AS/400 Feature is an ILE program, it cannot directly call the functions you need because it is not bound to your service programs.

5250 Screen Scraping

Question Can I use VisualAge for Smalltalk to revamp AS/400 5250 screens? Is there any restriction? What software do I have to use?

Answer Screen scraping and 5250 emulation interface driving are useful ways to put a GUI wrapper around your legacy AS/400 interactive applications. in VisualAge, this is accomplished through the Emulator High-Level Language Application Programming Interface (EHLLAPI, pronounced as ″EE-huh-lah-pee″).

EHLLAPI is a standard that is documented in the OS/2 EE 1.3 *EHLLAPI Programming Reference*. This interface is generic to all terminal emulations in the APPN/SNA environment, including the 5250 family (not just 3270).

In VisualAge for Smalltalk, the EHLLAPI protocols are implemented in the *Abt3270Hllapi*, *Abt3270Terminal*, *Abt3270Screen*, and *Abt3270HllapiError* parts. These parts have been tested and documented for use with 3270 applications, but the 5250 environment is so similar that some customers have already successfully used them in 5250 applications.

There are many products that include EHLLAPI with their 5250 or 3270 emulation. So, a feature was created for the *Abt3270Hllapi* parts that allows VisualAge application developers to configure the EHLLAPI DLL name and entry point name to use their favorite emulator.

To do this, you must create a *startup* class method in your application class (that is, *YourVisualAgeApp class>>#startup*). Within that method you should set the EHLLAPI DLL and entry point attributes like this:

Smalltalk

```
Abt3270Hllapi dllName: 'pcshll.dll'.   ″Substitute your EHLLAPI DLL″
Abt3270Hllapi entryPointName: 'HLLAPI'. ″ and its entry point here. ″
```

The above is the default which is set for the PC 3270 product. If you use an EHLLAPI product other than PC 3270 without the above two messages in your *startup* method, you will get an error message saying ″PCSHLL.DLL not found″.

This should work with any EHLLAPI product as long as it provides a DLL file that has one entry point with the following four parameters:

c

```
unsigned short far *  /* function number     */
char far *          /* data area          */
unsigned short far *  /* data length        */
unsigned short far *  /* position/return code */
```

It also must be a 16 bit (segmented addressing) function that uses one of the client communication routers supported for use with VisualAge for Smalltalk:

- Client Access/400 for Windows
- PC Support/400 for Extended DOS
- Communications Manager/400 (comes bundled with CA/400 for OS/2)
- Communications Manager/2 for OS/2
- Personal Communications/3270

Check the requirements section of your *VisualAge for Smalltalk Installation Guide* or *Resource Catalog* for specifics on supported levels.

A must-have reference for VisualAge EHLLAPI development is the ITSO Redbook *VisualAge: Building GUIs for Existing Applications*. Note that the example application is for a 3270 environment, but it can be easily extended to 5250 environments.

DDM Performance and File Access Parts

Question We try to access an AS/400 file with 500 fields and 80000 records from VisualAge for Smalltalk. We need to work with only 10 fields in the record and 1000 records. In order to access the database with DDM and instantiate the data, we need 5 minutes when we use the *Keyed File* part and the *readNextKey* method. I assume it is because VisualAge for Smalltalk always instantiates the whole record. Is there a solution to do this better? What is the difference between the *Sequential File* part and the *Direct File* part, anyway?

Answer One thing you might try in order to improve performance is to create a logical view (logical file) on the AS/400 that is based on this 80,000 record physical file. The record format for this logical file would contain only the desired fields. In this way you would only be bringing 10 fields per record over the communications line instead of 500 fields per record.

Also, since the file is a keyed file, you could use the *readAt:* method to access records of a particular key. This way you would not need to search sequentially (in key order) through the file to find the key you want.

Direct and sequential files are in fact very similar. However, here are some differences:

- Both direct and sequential files are created using the same CL command. However, after a direct file is created, an INZPFM command is issued to initialize the member with a large number (default = 10,000) of inactive records. This is not done for a sequential file.

- With a direct file, you can write a record at a specific location in the file. This is not possible with a sequential file.
- With a sequential file, you can do a *readAll*. This allows you to read all records of a file without opening the file. It uses the DDM ULDRECF command. This is not possible with a direct file.
- Finally, when you send repeated *readNext* messages to a direct file, the records are returned in relative record number sequence, whereas with a sequential file they are returned in the order in which they were written.

Migrating from OS/400 V3R0M5 to V3R1

Question My customer has just migrated OS/400 from V3R0M5 to V3R1. At this current level, VisualAge for Smalltalk cannot connect to the AS/400. That is, when we try to connect, a VisualAge for Smalltalk debugger reports that an ENVY exception has occurred. The following products are installed:

- Workstation
  - OS/2 Warp Version 3
  - VisualAge for Smalltalk and AS/400 Feature
  - Communications Manager/2 Version 1.11
- AS/400
  - OS/400 V3R1
  - Host Server feature (5763SS1)

We tried to install the QENVY library on the AS/400 but it doesn't help to solve the problem. What can we do?

Answer First, you do not need the QENVY library on a V3R1 system. Here are the things I would look at or try:

- Try executing *AS400System reconfigureAllSystems* before attempting to connect to the AS/400.
- Is your conversation security set up properly in CM/2? There is a known defect in CM/2 1.11 wherein our product will not work properly unless you set up the conversation security. To check (and if necessary fix) this:
  1. Start the Communications Manager Setup program (there should be an icon in your Communications Manager/2 folder).
  2. Select **Setup**
  3. Select the configuration you are using.
  4. Click on the **Additional Definitions button**.

5. Select the proper connection type in the Workstation Connection Type window and 5250 emulation in the Feature or Application window.
6. Click on the **Configure** button.
7. You should see a window entitled *Communications Manager Profile List*.
8. Select **SNA Features** and click on the **Configure** button.
9. Select **Conversation Security** in the features list.
10. Do you see anything in the Definition Comment list? If not, go to Step 13; otherwise, continue.
11. Is there just one entry in the Definition Comment list and is it an asterisk? If so, your Communications Manager program is probably okay. If not, continue.
12. If you feel comfortable doing this, I would delete all entries and continue.
13. You should be in the SNA Features List window. Select **Conversation Security**. Press the **Create** button. In the new window, select the **Utilize User Profile Management** box. Click on **Add**. Do not enter a user ID or password. When you are done, you should see an asterisk in the Definition Comment list.
14. Click on **Close** to exit the SNA Features List window.
15. Click on **Close** to exit the Communications Manager Profile List window.
16. Click on **Close** to exit the Communications Manager Configuration Definition window.
17. Close remaining windows.

Compound Key

Question Use the *readAt* method to retrieve records from a keyed file. When one record field is the key, this works fine, but when the key is made up of two fields, I′m not able to get it to work. The first field is a character field and the second field is a zoned field. How should this be put together?

Answer You can do this as follows:

```
Smalltalk
key := Array with: 'XYZ CORP' with: '12345.67' asDecimal.
record := file readAt: key.
```

Refer to *VisualAge for Smalltalk AS/400 Connection User's Guide and Reference Version 3.0* for a good discussion of this. Here is an excerpt from this manual that deals with keys.

Using Keyed Files

A file can have a key that consists of one or more fields. When a key consists of one field, it is called a *simple key*. When a key consists of more than one field, it is called a compound key. Data can be accessed based on the entire compound key or a subset of the fields in the compound key. When a key consists of a subset of the compound key fields, it is called a partial key.

Specifying a Key

Many of the file actions for a keyed file require a key as a parameter. The value of the key can take several forms depending on the complexity of the key.

Specifying a Simple Key

When a file has a simple key, a simple value can be passed to the actions requiring a key. For example, if the simple key is described as ZONED or PACKED, an instance of Decimal can be passed as the key value. If the simple key is described as CHARACTER, an instance of String can be passed as the key value.

In general, if the field can be described as a key, the corresponding Smalltalk object can be passed as the key value.

Specifying a Compound Key

A compound key can be specified in several different ways. If the required key value consists of all of the fields in the compound key, an instance of the recordDescription of the file can be passed as the key value.

A second approach is to pass the requested key values in a collection. For example, if a compound key consists of three fields and the desired key value is keyValue1, keyValue2, and keyValue3, respectively, a valid code fragment would be as follows:

```Smalltalk
aKey := Array
    with: keyValue1
    with: keyValue2
    with: keyValue3.

aRecord := aFile readAt: aKey.
```

Another example of a valid use of a collection as a key follows:

```Smalltalk
aKey := OrderedCollection new.
aKey
  add: keyValue1;
  add: keyValue2;
  add: keyValue3.

aRecord := aFile readAt: aKey.
```

The values in the collection are assigned in order to the respective key fields for the duration of the keyed operation.

Specifying a Partial Key

The partial key can be described as an AS400RecordDescription having only the desired key fields. The advantage to this approach is that the partial key record can be visually connected in the Composition Editor. The disadvantage is that if the field definitions change, fields would have to be changed in both the record description for the file and the partial key record.

A second approach is to pass the requested partial key values in a collection. For example, if a compound key consists of three fields and the desired partial key consists of the first two fields having the values keyValue1 and keyValue2 respectively, a valid code fragment would be as follows:

```Smalltalk
aKey := Array
    with: keyValue1
    with: keyValue2.

aRecord := aFile readAt: aKey.
```

The values in the collection are assigned in order to the respective key fields for the duration of the keyed operation.

Data Queue Program Temporary Fix

Tip

If your application uses OS/400 V3R1 data queues (with VisualAge for Smalltalk AS400DataQueue parts or E4DataQueue services), your application may hangup or exhibit other errors if PTF SF24792 has been applied to the host system where your application's data queues reside. See Authorized Program Analysis Report (APAR) SA47183 for more problem symptom details.

PTF SF24792 is included in cumulative fix volume C95276310 and is a prerequisite for PTF SF24769 (which is included in the latest cumulative volume C95304310).

The solution to this problem is to apply PTF SF26431. This PTF now supersedes SF24792, but it is not yet included in any cumulative volume, so it must ordered. (If you have ECS, use the SNDPTFORD CL command.) Be careful to follow the cover-letter instructions.

As a circumvention, you can remove PTF SF24769 and PTF SF24792, being careful to follow the PTF cover-letter instructions for their removal. If you choose to do this, you should reapply SF24769 after SF26431 has been applied.

Record Descriptions

Question

Is it possible to define a completely new record description in VisualAge for Smalltalk (including the host details) and then have VisualAge propagate the host information to the AS/400 to generate the DDS?

Answer

For physical files, yes. Assuming the record description (including host field name alias), system, library, and file name settings are already set to valid values, then just connect whatever is the significant event in your application to the AS/400 file part's *create* action. The file and its DDS will be created on the specified host when that event occurs.

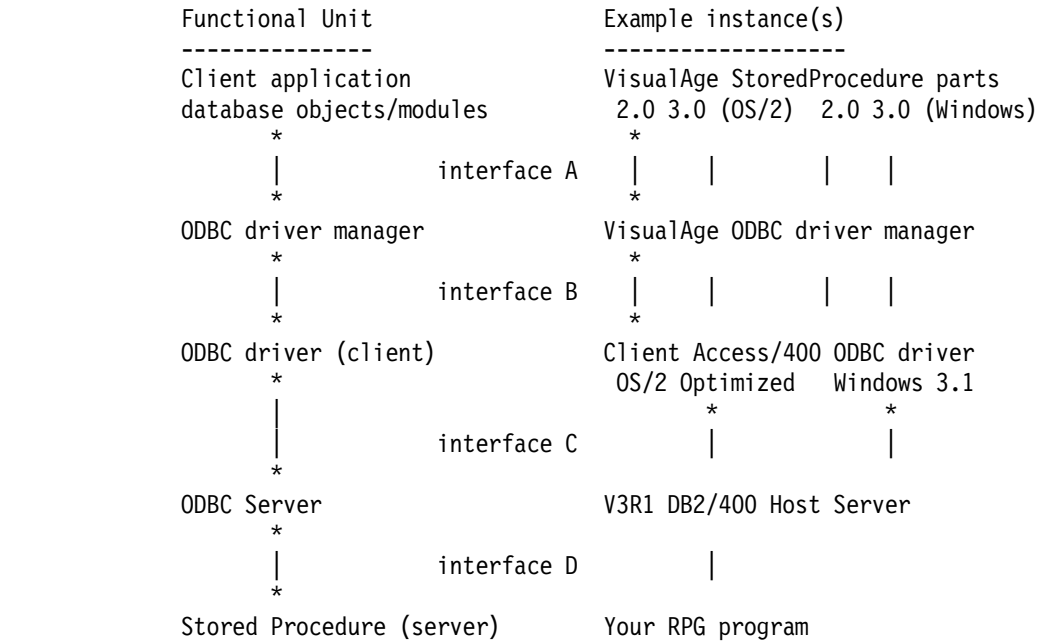
For logical files (or DB2/400 views), no. To create logical files from within your VisualAge for Smalltalk client application, you would need to imbed your DDS source in an argument of a remote command to first create the DDS source file member and then send a CRTLF remote command using this newly created DDS source file.

Alternatively, DB2/400 views can also be constructed through ODBC SQL statement execution.

Stored Procedures in RPG

Question We want to use the VisualAge for Smalltalk ODBC support to access the AS/400 database. Is it possible to create stored procedures on the AS/400 with RPG and call those procedures through the *ODBC stored procedure* part under Windows and OS/2?

Answer Yes, under some circumstances, *Stored Procedure* parts can be used with ODBC instead of RPC parts. The main constraint is compatibility of the interfaces between the five levels of the ODBC execution of the stored procedure:



Each interface can pass certain argument objects in an agreed format through agreed channels. Some of these interfaces support passage of 0, 1, or multiple arguments for each input only (IN), output only (OUT), or both (INOUT). Output parameters are also referred to as *result sets*.

DB2/400 supports RPG stored procedure calls, as well as C, FORTRAN, REXX, CL, and several other program language bases. See the *DB2/400 Database Programming Reference* and *SQL Reference*

Proper Exiting to Remove Dependents

Question I have noticed that two classes, AS400APPCConfiguration and E4AS400Broker, have left instances around after executing my program. Is there a proper way of exiting a program that uses the AS/400 connection so that the above two classes will not leave any instances around? Should I be concerned about the instances? Can these instances get included in the image when I package?

Answer While you are in development mode, there should be no problem. You should be refreshing your image's configuration model every time your image is started. Be sure, in your application class *startUp* method, to do:

```
Smalltalk
AS400System reconfigureAllSystems
```

This also should be done whenever your communication configuration is changed or restarted while the application image is still active.

If your application must have a default system, this default should also be set in the *startUp* method, whether by prompting the user or by loading this from your application's configuration data.

If you do nothing about these before packaging your development image, you will leave instances in the packaged image. However, if you do the appropriate things in your application class #startUp method, as described above, these instances will be removed from the configuration model and and BrokerRegistry, eventually to be garbage collected. Thus, they are of little consequence other than the space and time overheads they cause by being left in the deployed application image.

To be conscientious and tidy about your packaged images, you need to create a *prePackagingActionsFor:* class method in your application classes to automate the image housecleaning. Browse implementers of prePackagingActions* for examples. This should include setting your application's configurable variables to nil (if they are lazy-initialized or are loaded from an object store at startup) or the application default.

The method E4FrameWorkApp class>>prePackagingActionsFor: anImage does "E4Service shutDownAll" to ensure that no active conversations are left in the image. To be sure that no global reference to your development server systems persist (preventing garbage collection),

1. Close all active AS/400 parts.
2. Clear all your application caches.
3. Sign-off all systems (*AS400System availableSystemNames*).

Generally, put your application in the pristine state that you would like it to be for a fresh-out-of-the-box application.

RPC Parameter Problem

Question I tried to use RPC with no arguments and I've got an error (MCH0802) because of the wrong number of passed parameters. On the AS/400, I saw that the problem was caused by the progrm QEVYMAIN which tried to call the program QIWS/QGYSETG using 9 instead of 7 parameters. How do I get the correct parameter list?

Answer It sounds as though you have a down-level version of QGYSETG. The Version 3.0 manual has a list of prerequisites for Version 3.0 to run. Please make sure that you have installed the necessary PTFs:

- SF23309
- SF23828
- SF23829
- SF24428

Variable-Length Data Queue

Question I defined an AS/400 data queue and I want my VisualAge for Smalltalk program to read from it. There is a partner program on the AS/400 that writes some data in that queue. The partner program does not always write a fixed number of characters, so that one time the message in the queue can have 100 characters, the next time it is only 50 characters.

If I define a record to read 100 characters, the VisualAge for Smalltalk application encounters an error when reading the 50-character message. The error comes from the at: method of ByteArray class. VisualAge for Smalltalk tries to read 100 characters, while the stream coming from the AS/400 has only 50 characters. If I define a record structure with 50 characters, I lose information from the 100-character message. Am I missing something?

Answer If you described the field data as (CHARACTER length: xx) this is essentially a fixed-length description. You have two options:

1. Have the partner program pad the messages with blanks. This is probably the easiest solution. There is a slight performance cost, in that the padded blanks will be translated, but if it's only 50 bytes, the cost should be minimal.
2. Use varying-length data description. This is a bit tricky because the partner program would have to include variable-length information in the data queue entry. To do this, specify *varying* for the field. If you are using the AS400RecordDescription on the settings view, open the settings for the CHARACTER field and go to Size page and check the variable-length check box. The resulting type will be changed to (CHARACTER length: xx) varying where xx is the maximum length.

   Variable-length data is encoded with two leading bytes of length information. This means your partner program would have to put length information on the data queue like this: LLDDDDDDD..DD where LL are the two length bytes and DDD is your data bytes. Do not include the length bytes in your data description length.

   For example, if your messages have a maximum length of 100, the field should be described as (CHARACTER length: 100) varying. The partner program should put the two length bytes and then all of the data (up to 100 more bytes) on the data queue.

Saving Image Causes Communications Problem

Question I am using VisualAge for Smalltalk, Version 3.0 for Windows, and I am connected to an AS/400. When I save the image and after that try to reaccess the AS/400, I get an error like: *Unable to connect. Router and communication DLLs are not available.* Can you help me?

Answer There is a problem in VisualAge for Smalltalk, Version 3.0 for Windows, when you save the image and then make any other APPC accesses before you exit VisualAge. VisualAge releases the APPC router DLL before the image is saved to avoid saving an invalid handle. When you then continue to access your AS/400 parts, you get the messages you mentioned.

One way to work around this problem is to exit VisualAge for Smalltalk immediately after you save the image. We have also provided a circumvention with the Version 3.0 CD-ROM which is referenced in the following excerpt from the readme.as4 file (also on the CD-ROM):

README.AS4

Known Problems and Testing Notes:

- Lost AS/400 Systems when saving image (Windows only)
  - Problem:

    When an image is saved under Windows, the available AS/400 systems may be lost.
  - Fix:

    To fix this, you need to file in AS4APPCW.ST from the readmefileins subdirectory into your Windows image.

RPG Program Calls

Question Can RPG programs be called within a VisualAge for Smalltalk application? If so, can the parameters be passed to and from the RPG program?

Answer Yes. An AS/400 RPG program (or any *callable* AS/400 program, regardless of the programming language used to create it) can be invoked by a VisualAge for Smalltalk client application. Read ″Chapter 4. Calling AS/400 Commands and Programs″ in the *VisualAge for Smalltalk AS/400 Connection User's Guide and Reference*, Version 3.0, which can be found in the online references on the VisualAge for Smalltalk v3.0 CD-ROM.

As for the passing of data between client and host server programs, there are several alternatives:

- AS400RemoteProcedureCall arguments (both input and output)
- Data queues (FIFO, LIFO, and keyed)
- Data areas (character and decimal)
- User spaces
- AS/400 database files

If the data is simple, RPC argument passing would suffice. Another consideration is whether or not your RPC (your RPG program) can run asynchronously, so that some concurrency of client and server applications would give better performance. Although it is possible to communicate with asynchronous RPCs through arguments, it might be easier to manage asynchronous RPC invocations by using one of the more loosely coupled alternatives, such as data queues.

Personal Communications for AS/400

Question Does Personal Communications for AS/400 for Windows 4.1 work in place of the Client Access router for the VisualAge for Smalltalk AS/400 Connection feature?

Answer No. While the Personal Communications product may provide AS/400 support, it does not provide an implementation of the E32APPC DLL, which is currently required by the AS/400 Connection feature.

Repeated Record Structures

Question We are investigating how best to return complex data structures back to a VisualAge for Smalltalk client. Our initial investigation has led us to the following conclusions:

- When using AS/400 Record Descriptions, there is no way to define a repeating structure for the record as a whole, or for a user-defined field within the record (a nested record).
- User spaces are the preferred means to achieve a repeating structure, at least for a simple record. (I′m not sure how to handle repeating nested records here.)

Am I correct?

Answer Repeating structures are supported, although not documented (sorry). This function is not available at the Composition Editor, but is available “under the covers.” Using this technique will probably render your record description useless within the Composition Editor, but will provide the function you are seeking.

The key to using repeating fields and repeating structures is the use of the ARRAY datatype. The ARRAY datatype allows collections of like types to be described. For example, if a RPC returns 100 fields of packed decimal numbers you *could* describe the record description as:

```
Smalltalk
(RECORD subclass: #DoNotDoThis)

  field: 'field1' type: (ZONED length: 4);
  field: 'field2' type: (ZONED length: 4);

 ⋮
  field: 'field100' type: (ZONED length: 4);
  endRecord
```

But using ARRAY, you could do this:

```
Smalltalk
(RECORD subclass: #DoThisInstead)

  field: 'fields' type: (ARRAY of: (ZONED length: 4) occurs: 100);
  endRecord
```

and the #fields getter method would return an array of decimals.

The ARRAY>>of:occurs: accepts any valid AS/400 type, so user-defined records could be used as well.

In many cases, a fixed number of array elements cannot be used because the results vary from request to request. In this case, you can describe the number of array elements based on the value of another field (defined before the array field). For example, if your RPC returns the detailed records for a given customer/order number, then you could describe the result as:

```
Smalltalk
(RECORD subclass: #VariableRepeatingEntryRECORD)

 field: 'numberOfEntries' type: (BINARY length: 32);
 field: 'entries' type: (ARRAY
                    of: MyOrderDetailRecord
                    occurs: 100
                    dependingOn: 'numberOfEntries');
 endRecord
```

where 100 is the maximum number of detail records that would ever be returned. In this case, the number of order detail records is determined by the value in the numberOfEntries field. The RPC program would need to fill in this value prior to returning the record.

To make these changes to your RECORD subclass, you will need to carefully edit the class definition within a Smalltalk browser and save the changes. You must include the #endRecord statement or your image will get corrupted.

In answer to the other part of your question, the User Space part does provide automatic implementation of variable-length length repeating stuctures when the requested byte range exceeds the length of the user-space data description. In fact, our implementation uses the ARRAY technique described above. To handle repeating nested records, simply describe the record within a record as described in the user guide references above, assign your record to be the #dataDescription of the user space, and you get the repeating support for free.

Simultaneous DDM Access

Question Is it possible to make two (or more) simultaneous DDM requests (for example: a readAll and a readNext), from two (or more) different Smalltalk processes to the same or different AS/400 file parts?

Answer No. The DDM data streams share one conversation between the client process(es) and the single DDM server job. See ″Appendix A. Guide to Jobs Started by the AS/400 Connection″ in the *VisualAge for Smalltalk Version 3.0, AS/400 Connection User's Guide and Reference* on the CD-ROM.

While the DDM architecture does support interleaving and multileaving of transaction data streams within a shared conversation, the contention between the client processes, in this case, is for the conversation state, which is a much lower-level resource, beyond the control of the public application-accessable protocol in the AS/400 Connection feature.

As a result, it is up to the (cooperating) concurrent client processes to ensure that only one DDM transaction is taking place at any time. This means that you need to use semaphores to enforce mutual exclusion around file-access operations, as you currently are doing. For more information, see ″Chapter 15. Processes and Synchronization″ of David Smith's book, *IBM Smalltalk, the Language*.

Synchronous Processing

Question Is it possible to issue a synchronous RPC, running on a separate process, and use this RPC to:

- Run an RPG program in the AS/400 in order to read the registers on a file and write on a queue, or in a user space?
- Read the queue or the user space using the VisualAge for Smalltalk parts, all of this simultaneously with other accesses to the same or different file using the AS/400 file parts, from the user interface process?

Answer Yes, you could start an AS/400 RPG program that would do the same kind of polling for changes of the database file that you currently do with the #readAll message from your client application. The RPC could be started with either #invoke or #invokeAsynchronous, depending on your client application′s need for control.

Synchronous calling would allow the termination of the RPG program to control the termination of the client process that called it. Asynchronous calling would allow the same client process to continue on, reading the data queue entries produced by the RPG program, and terminating under its own control, depending on the data queue or user space contents. Asynchronous calling also has the advantage of requiring only one process and simpler coordination. By contrast, synchronous calling requires a separate process to read the data queue or user space and coordination with the RPC-calling process.

You have two other alternatives:

- Your AS/400 program that processes the event and results queues could write an entry to a data queue or user space after it has made an update to your production floor-model database file. This would signal a client process waiting to read the queue that the client model needs to be refreshed. Notification would elminate the need for polling and the data queue entry content could be used for more selective record retrieval, depending on what was actually updated in the file.
- The AS/400 file (assuming this is OS/400 Version 3R1 or higher) can itself put entries in a data queue or user space to signal its update. This can be done by attaching DB2/400 file or field update triggers that directly invoke data queue APIs or another program to create the data queue entry. This has the advantage over the previous alternative of not impacting the event-and-result queue program, but the file DDS is more complex.

One other consideration:

Any asynchronous file access initiated by a user interface event is subject to the same mutual exclusion constraints that were mentioned above. If the user interface process can wait for a pending #readAll to complete, then it would be sufficient to bracket the user interface file access within the #critical: argument block for the same mutal exclusion semaphore that guards the #readAll file access. However, if this produces any unacceptable user interface response, you can implement a two-level priorty queue as follows:

Smalltalk

```
| highPriorityDDM lowPriorityDDM |

highPriorityDDM := Semaphore forMutualExclusion.
lowPriorityDDM  := Semaphore forMutualExclusion.
...
"The code that polls the AS/400 file must pass both mutex guards."
[ [ true ] whileTrue: [
  lowPriorityDDM critical: [
     highPriorityDDM critical: [
        myRecords := myFile readAll.
        ]
     ]
  "Do all non-DDM processing of myRecords outside of the
   nested critical sections."
    ...
  ]
] fork.
...
"The code that accesses the AS/400 file within an event
 handling block that is triggered by a user interface event
 need only pass the highPriorityDDM mutex guard.  This allows
 it to sneak into DDM ahead of pending lowPriorityDDM accesses.
 However, this will not interrupt a #readAll that is already
 doing DDM access."

 highPriorityDDM critical:
    [ userInfo := myFile readAt: infoKey ].
```

ODBC Driver and Host Variables

Question I am using VisualAge for Smalltalk on Windows 3.11 with CA/400 and I am trying to use the ODBC driver that comes with Client Access. When I query DB2/400 through a multiple-line query, every thing works fine. When I use a host variable I get an error: *IM001* {*microsoft ODBC DLL*} *driver does not support this function.* Any ideas?

Answer ODBC does not support host variables. Use parameter markers instead. Refer to ″Appendix E. Comparison Between Embedded SQL and ODBC″ in the *Microsoft ODBC 2.0 Programmer's Reference*. There it says, ″ODBC uses a parameter marker in place of a host variable, wherever a host variable would occur in embedded SQL.″

Asynchronous RPC with Large Arguments

Tip To all VisualAge for Smalltalk application developers who use AS400RemoteProcedureCall>>#invokeAsynchronous

A problem occurs when any RPC argument is larger than 256 bytes. There are two symptoms:

1. On the AS/400 host, in the user's main server job log (use WRKUSRJOB CL command and ″Work with″ Option 5 to see), there is a CPF2498 message in the QIWS/QEVYMAIN program saying that the data queue entry it is sending has an ″Invalid length.″
2. At the same time, the user's auxiliary server job (which was started by the main server job to execute the asynchronous RPC) is hung up, waiting to receive the data queue entry from the main server job, which never gets there. You can display the auxiliary server job's call stack to verify this (″Work with Job″ Option 11).

From your client application's view, depending on your configuration and use, the application may hang up, receive a time-out exception, or endlessly retry AS400RemoteProcedureCall>>#resultIsReady.

APAR SA55627 describes a temporary workaround.

Data Queues

Question I'm looking for a way to share a data queue (in QTEMP) between a VisualAge for Smalltalk application and a native AS/400 application. In normal operations, each application runs in a different job. How can I make an RPG program, that reads lots of data, summarize the data and put the results (multiple lines) into a data queue? When I try, the VisualAge for Smalltalk program then handles only summarized data.

How do I run both applications in the same job? Or is there another way of handling this situation?

Answer By definition, each job has its own QTEMP library and the contents cannot be accessed by other jobs. Since the data queue in VisualAge for Smalltalk uses its own job, and your native AS/400 program runs in another job, the solution is to use a different (permanent) library such as QGPL. It is not possible to run your native AS/400 program and the VisualAge for Smalltalk data queue server program in the same job.

File Open and Close

Question In order to benefit reuse, I keep my applications separate from the RDBMS-handling. When using the AS/400 feature, this means that I never use a file in an application, but that objects request services from an AS/400 database application. The biggest problem with this approach is that the application can not control file opens and closes. This means I have to open and close the files with every transaction.

To avoid this (performance) problem, I wanted to open the necessary files in the beginning of an application, followed by an OVRDBF SHARE(*YES), as we do in a native application where a lot of calls are executed. For these commands I use the AS400RemoteCommand class.

The problem is that these commands are executed in another job on the AS/400 rather than the job where all DDM requests are handled. Therefore, this approach does not work. How should I tackle this problem?

Answer Try using the file part action *remoteCommand:*. This will run in the DDM job. Depending on the OS/400 version you may not be be able to get this to work as the override scope (OVRSCOPE) must be specified as *JOB on the OVRDBF command. I know this is available on V3R1, I'm not sure about V2R3, and I know it is definitely not on V2R2.

Sign-On Screen

Question I'm having a problem with the AS/400 sign-on screen in a packaged application under VisualAge for Smalltalk version 2. In edit time it works fine, but in the packaged application, all the text has been replaced with AS400BaseRun message errors because the correct text seems to be missing. What should I do to get my messages back?

Answer Your problem is that you do not have the AS4RTE20.MRI file in the same directory as your packaged image. You can find AS4RTE20.MRI in the RUNTIME subdirectory of the VisualAge for Smalltalk directory hierarchy on your server. That is, put X:\VISUALAG\OS2\RUNTIME\AS4RTE20.MRI, where X: is the drive letter for your VisualAge for Smalltalk server.

If you want to solve the problem for future client installs, you should copy the .MRI file to the ABT subdirectory of your VisualAge for Smalltalk server directory.

Error Using #readAll

Question I have run into an error when using the #readAll method for E4KeyedFiles. The method works fine initially, but I receive an "Invalid usage" error when I have another instance of a keyed file open for the same file. Is this a bug? I understand that I can't use *readAll* if my *E4KeyedFile* is already open, but this is another instance and should be independent of the file that I am using for *readAll*. What do I do wrong?

Answer This is not a bug. As far as the DDM Server is concerned, the file is open. We do not start a new logical session for each agent unless you are going to a different AS/400.

DDM and RPC

Question How can I force an AS400RemoteProcedureCall to be executed in the same VisualAge for Smalltalk for AS/400 job where DDM requests are handled? The goal is to have an RPC execute the command OVRDBF xx OPNDBF SHARE(*YES) and DDM open and close files per transaction for optimal reuse of code.

Answer What you are looking for can be accomplished by using remoteCommand: which will execute commands through the DDM job. This is a piece of code that I wrote:

```Smalltalk
| file |

file := AS400DirectFile newPart

    recordDescriptionName: #TestRecordDefinition;
    filename:'TESTFILE';
    libraryName:'TESTLIB';
    yourself.

file remoteCommand:'ADDLIBLE TESTLIB'.
```

Use this code and send an OVRDBF as the remote command. You could also issue a CALL PGMX this way, however, you would have to find another way to retrieve any returned information, possibly through a data area.

ODBC Default Library

Question When using the ODBC driver to connect to the AS/400, there is a defined default library. How can I change this when I am about to log on to the AS/400?

Answer You can set a default library list in ODBC.INI. Thus, if the proper library name (or names) is in the ODBC.INI on your machine, then the Client Access ODBC driver should find the table.

File Locking

Question If the file lock option is set to *exclusive*, does that mean that the entire file is locked or just the current record? Also, if the option is set to *Share with readers*, does that mean the entire file is locked from updaters?

Answer When the file access mode is set to exclusive, the file is locked in such a way that neither readers nor updaters can access the file. When the file access mode is set to readers, the file is locked in such a way that those who open for read-only access can read the file, but those who want to open for update access cannot.

Unique Keys

Question When using a keyed file where the key or keys are specified as unique and I try to write a new record with a key that already exists in the file, is there a way to get a return code of this being a duplicate record, or is that where an exception handling comes in and an excection would be generated?

Answer This is where you want to use exception handling. Here is a script that will demonstrate the con cept of exception handling for the particular case you cited.

Smalltalk

```
exceptionOccurred: aCollectionOfMessages

"Check to see if expected messages were detected.
 This will check for a nonexistent message for example
 purposes only.  This script should be used with an
 event-to-script connection for the exceptionOccurred event."

| aMessage |

aMessage := aCollectionOfMessages
     detect: [ each | each identifier = #E4DUPKSIRMY
           ifNone: nil].
^aMessage notNil
  ifTrue: [
     CwMessagePrompter warningMessage:
       'Duplicate key, will not add to file.'.
     #ignore ]
  ifFalse: [
     #cancel ]
```

You can try this out by connecting the *exceptionOccurred* event of the *AS400KeyedFile* part to this script.

Logical Record Format

Question I retrieved the following record description for physical file PART:

```
Smalltalk
(RECORD subclass: #PART)

  field: 'prtnbr' hostField: 'PRTNBR' type: (ZONED length: 6);
  field: 'prtdsc' hostField: 'PRTDSC' type: (CHARACTER length: 30);
  field: 'grpnbr' hostField: 'GRPNBR' type: (ZONED length: 6);
  key: 'prtnbr';
  endRecord
```

I then repeated the same operation for logical file PARTL1 based on PART and got:

```
Smalltalk
(RECORD subclass: #PARTL1)

  field: 'prtnbr' hostField: 'PRTNBR' type: (ZONED length: 6);
  field: 'prtdsc' hostField: 'PRTDSC' type: (CHARACTER length: 30);
  field: 'grpnbr' hostField: 'GRPNBR' type: (ZONED length: 6);
  key: 'grpnbr';
  key: 'prtnbr';
  endRecord
```

Because both files share the same field names, PARTL1 should inherit from PART. Consequently I changed the preceding class definition to the following:

```
Smalltalk
(PART subclass: #PARTL1)

  key: 'grpnbr';
  key: 'prtnbr';
  endRecord
```

However, when I try to read PARTL1, I get a walkback saying that *hostFieldNames* is nil.

When a logical file includes all the fields of its physical file, how do I make the logical file's record format a subclass of the physical file's record format?

Answer You get a walkback saying *hostFieldNames* is nil, because the hostFieldName class variable is inherited from PART, but the value it holds (the collection of host field names) does not get copied to the subclass.

The hostFieldName array is constructed by #field:hostField:type: invocations and assigned in the RECORD class>>endRecord method.

The RECORD class is intended as a template for creating AS/400 record classes. If you are intending to define new views of an existing record using a subset of its fields, alternate keys, or both, then the #logicalView method would be a more straightforward alternative. (See the "Comments" text of RECORD class>>logicalView or STRUCTURE>>logicalView for examples.)

It should be noted that #logicalView cannot be used to implement a client-side "logical join" view of two or more record classes. To do that, you must implement the view or logical file on the AS/400 and then derive the client record description from this new view, as you did in your first implementation of PARTL1.

Inheritance from subclasses of RECORD can be used to aggregate additional attributes. But this also might be better handled by having an instance variable of the new class to contain an instance of the record. It would depend on how much of the record's protocol needs to be visible outside of the new class in your application. And, if you use inheritance, be sure to copy the record's class variables (such as hostFieldName) to the subclass.

Fastest DB2/400 Access

Question Is it true that ODBC is the fastest way to access AS/400 data?

Answer If you are using SQL access exclusively, yes, ODBC is the fastest access to AS/400 data, assuming the following:

- You are using the VisualAge for Smalltalk ODBC driver manager.
- You are using the V3R1 Client Access/400 ODBC driver. The V3R1 DB2/400 host server has been optimized for use with this ODBC client driver. The V2R3 PC Support ODBC driver has a different, more

restricted architecture. I have no benchmarks for other AS/400-operable ODBC products.

- You are not using the E4SQLAccess services. These DRDA-based service classes are intended only to ease the migration of previous ENVY/400 customers into VisualAge for Smalltalk They will still be available in the next release of the VisualAge for Smalltalk AS/400 Connection but they will be phased out. Do not use them for new development.
- You are talking about performance with respect to a particular client/server conversation—that is, a single server, not a distributed database on multiple servers. I have not seen benchmarks on multiserver applications for ODBC, so I wouldn't know.

ODBC is also the recommended choice for portability, interoperability, and forward compatibility. However, if the AS/400 is your only application server platform and you are not limited to SQL access, you can get faster response and (for simple retrieval or update) higher throughput by using the DDM-based AS/400 database services provided in the VisualAge AS/400 Feature (AS400KeyedFile, AS400SequentialFile, and AS400DirectFile parts).

In an OS/2 client environment, DB2/2 with DDCS/2 is also a viable choice. However, some customers have found that DDCS is slower than ODBC in their single-server environment. This may be because the DDCS transactions are routed through an additional communications server.

This discussion would not be complete without a "your mileage may vary" disclaimer: Your application constraints, the complexity of the queries, the database topology (in a distributed multiserver database), the communications media and its configuration, and the processing environment (system load, resource availability, and so on) can all influence actual performance in a way that might favor one of the alternatives to ODBC.

This being said, your best choice will be made by testing the alternatives in a small prototype of your product's typical or most critical environments.

Access OV/400 Document

Question Is it possible to access (read/store/change) AS/400 documents through the VisualAge for Smalltalk classes?

Answer We do not support manipulation of documents except by means of CL commands and System APIs. You can access CL commands using the AS400RemoteCommand part and System APIs using the AS400RemoteProcedure. Using CL commands, you could dump the contents of an OV/400 document to an outfile. You could then read the outfile using the AS400SequentialFile part.

The meat of an OV/400 document is stored in Revisable Form Text (RFT) or Final Form Text (FFT) format. Even if your goal is only to bring back plain text from a document, this is nontrivial as the files (theoretically) ca n contain margin text, backspaces (which essentially negate the previous character), and other text controls that are either difficult to deal with or non-ambiguous. The RFT is especially difficult.

AS/400 Connection through TCP/IP

Question Can I use the AS/400 feature (DDM, remote procedures) over TCP/IP without the Client Access router?

Answer VisualAge AS/400 Feature supports only APPC/APPN communications. This is true for both Version 2.0 and Version 3.0. Client Access for OS/400 Optimized for OS/2 supports APPC/APPN communications via TCP/IP. Client Access for OS/400 (V3R1M1) for Windows 3.1 also supports APPC/APPN via TCP/IP. This product does not ship a TCP/IP stack, however; you would have to purchase one separately. But, in either of these cases you'd have to purchase Client Access.

Watch this!

If you are running under OS/2 and have an APPC/APPN connection through Communications Manager for OS/2 (CM/2) or Communications Manager for OS/400 (CM/400)), you do not need Client Access unless you are in a DBCS environment.

AS/400 Feature and TCP/IP

Question Is it possible to use the AS/400 Feature communicating directly to the AS/400 using TCPIP without having Client Access/400 or PC Support/400 loaded on the client machine? If so, would I need the Communications Feature to communicate directly to the AS/400 from a VisualAge for Smalltalk application?

Answer Yes, the client application can talk to an AS/400 across a TCP/IP-based connection through the VisualAge for Smalltalk communications feature, provided you have client software that supports the TCP/IP stack. You could do things like simple file transfer (FTP), remote job submittal, and the usual services that you get with TCP/IP. Your application will have to manage those services.

However, if you want to use the VisualAge for Smalltalk AS/400 Feature parts (which shield the application programmer from many communications and service management details), then the client application must have an SNA APPC protocol stack available to talk to the AS/400 host server programs. If you need to use a TCP/IP connection to get there, you can still do it, if you have Anynet client software. Anynet provides the APPC stack and wraps it in an IP conversation. This is transparent to the VisualAge for Smalltalk client application and the host server programs.

Anynet support comes with Client Access/400 for Windows, and both CM/2 and CM/400 for OS/2. If the VisualAge AS/400 feature with Anynet satisfies your application′s needs for AS/400 services, then you don′t need the VisualAge Communications Feature (unless, of course, your VisualAge application also talks to other IP-based non-AS/400 servers, or uses 5250 terminal emulation or screen scraping for legacy AS/400 applications).

Remote Procedure Call Messages

Question Is it possible to retrieve messages sent to a remotely called AS/400 program? When I do the following:

```
Smalltalk
^(AS400RemoteProcedureCall new
  procedureName: 'MYRPC';
  libraryName: 'MYLIB';
  arguments: (MYPARM new parmcde: 'A'))
     invoke;
     result
```

the answer is an E4Result object with three instance variables:

- *result*
- *status*
- *messages*

Variable *result* contains my updated argument record, *status* contains #SUCCEEDED and *messages* contains an empty ordered collection. Yet I know for a fact that the invoked program MYRPC had messages sent to it while running. How can I retrieve them? (Ideally the E4Result object would contain an ordered collection of E4ErrorMessage objects.)

Answer

What you are trying to do is not possible using VisualAge for Smalltalk Version 2, but it is possible using VisualAge for Smalltalk, AS/400 Connection, Version 3. I just created a window that allows you to type in any CL command. It issues an RPC to QSYS/QCMDEXC which issues the CL command. Then it displays a list of the message identifiers of all of the messages in the job log that have message identifiers. I have a pop-up menu for the list that allows me to show the message text for any message identifier.

Here are the *jobMessages* and *showMessage* methods:

```
Smalltalk
jobMessages

 | jobLog messages |
 (jobLog := AS400JobLog newPart)
   system: (AS400System named: #RCHAS400);
   openContents.
 messages := jobLog contents reject:

 jobLog closeContents.
 ^messages
```

```
Smalltalk
showMessage: anAS400ProgramMessage

 (self subpartNamed: 'Message Prompter1')
   messageString: anAS400ProgramMessage messageText;
   prompt
```

Although we only support retrieving all messages from the job log, since each Smalltalk program message object has a unique identifier within the job log (even if there is more than one message with the same AS/400 message identifier), you could simulate getting only new messages by keeping track of which message identifiers you've seen and rejecting messages you've already seen.

Remote Command Fix

Question After installing VisualAge for Smalltalk Version 3.0 for OS/2, we have a problem when using the AS400RemoteCommand. The command is not performed because of a dump of program QGYSETG in library QGY. We are running OS/400 version 3.10. The beta version of VisualAge for Smalltalk Version 3.0 was running correctly.

Do we need a PTF for the AS/400 or for any VisualAge for Smalltalk parts?

Answer You may need PTF SF24428 which supplies additional function in QGYSETG for optimal APPC connection in the Host Server List APIs (see APAR SA44719 for details).

This PTF is listed along with other needed PTFs in the "Prerequisites for OS/400" section of chapter 1 in the new "VisualAge for Smalltalk, AS/400 Connection User's Guide and Reference." The other PTFs are SF23309, SF23828, and SF25419 (which supersedes the listed SF23829).

Fixpack Problem

Warning

A problem was found in fixpack ABTTA43A.EXE. This especially affects VisualAge for Smalltalk, Version 3.0, application developers working on a Windows platform.

If you have already installed AS/400 Connection fixpack 1 from ABTTA43A.EXE in a Windows client image, it is recommended that you reinstall the Version 3.0 AS/400 Connection feature, and then install the AS/400 Connection fixpack 3 from ABTTA43C.EXE.

Do not install ABTTA43A.EXE if you have a copy and have not already done so.

The new fixpack (ABTTA43C.EXE with accompanying ABTTA43C.TXT file) is available in the usual VisualAge for Smalltalk fix directories at:

ftp://ps.boulder.ibm.com/ps/products/visualage/fixes

For those with Internet FTP access but no Web browser, get the ASCII file, ABTTA43C.TXT by anonymous FTP, switch to binary transfer, and then get ABTTA43C.EXE. Follow the ABTTA43C.TXT instructions to install the AS/400 Connection fixpack 3.

User Profile Name

Question Is there a way to figure out the name of the AS/400 user profile that is signed on with the VisualAge for Smalltalk application on the AS/400?

Answer Try the following:

Smalltalk

```
E4CommunicationService UserID
```

Referential Integrity Constraint Violation

Tip

A problem has been found when a VisualAge for Smalltalk, Version 3.0, application violates a referential integrity constraint that is defined for an AS/400 table.

The symptoms are:

- The user's server job's log (in subsystem QCMN) has CPF503A message.
- The exception is not signaled to the client's handler.
- The exception message is written to the Transcript or (for a packaged client) the system error log.

This problem happens because the severity of the error that triggers exception signaling is interpreted differently in the Version of DDM services used in version 3.0 of the AS/400 Connection feature. There are a few exception conditions that should always be signaled, but their DDM severity code does not always reveal this. CPF503A is one that we missed.

The general workaround is to add the exception's message ID to the collection of message IDs for exceptions that should always be signaled. To do so, create a scratch edition of the E4AS400FileService>>promotableMessages method as follows:

Smalltalk

```
promotableMessages

"E4AS400FileService>>promotableMessages with added exceptions:
    CPF5027 (record locked exception)
    CPF503A (referential constraint violation) "

    ^#( CPF5009 CPF5035 CPF5027 CPF503A )
```

After you create this scratch edition of #promotableMessages, your exception should then be signaled so that your exception monitoring should be able to catch this exception.

Reestablish Connection

Question If my AS/400 goes down because of a power outage or an IPL, my VisualAge for Smalltalk application results in a severe ENVY error, when it tries accessing the AS/400 by opening, reading, or signing on. The only way to resolve this is by booting the PC. However, I want to systematically reestablish a connection with the AS/400 without having to reboot the client. How can I resynchronize the AS/400 and the PC?

Answer At the point where the loss of connection is detected (probably in your exception handling for whatever AS/400 part you were accessing at the time), do the following:

Smalltalk

```
"Close all services for the failed system connection by using
 the #signOff method.  During #signOff, additional exceptions
 can be signalled, but ignore them like this..."

 myAS400System signOff Y onExceptionExecute: [ :aSignal | ].
```

Do whatever application cleanup is needed to put the client side of the transaction in a resumable state. This might be a good time to alert the user and prompt them to either end the application or have your recovery code wait for the server to become available.

If the user or your recovery code chooses to wait on availability of the failed system, you need to do some kind of polled waiting loop to reestablish the connection.

Complex applications with multiple open AS/400 parts and/or asynchronous processes benefit by this use of AS400System protocol (#signOff and #signOn) because any part that is sensitive to a server connection's failure or recovery can be hooked to the appropriate AS400System event (for example, #aboutToSignOff). In this way, the part can control its own shut down and recovery.

Application Packaging

Tip There is a problem when packaging images using the AS/400 Connection parts in Version 3.0.

The problem stems from the fact that two pool dictionaries are being removed by the packager, resulting in walkbacks when accessing local message text (for instance, as the result of a communications exception). The fix is to force the packager to include these dictionaries in the packaged image. This can be accomplished by adding the following method to the application class of one of your applications to be packaged into the final run-time image:

Smalltalk

```
packagingRulesFor: aPackagedImage

aPackagedImage
  includeGlobalVariable: #E4BaseNlsSeparatedMessages;
  includeGlobalVariable: #E4SqlNlsSeparatedMessages
```

File Access through Library List

Question How can I access keyed files without qualifying the library but using the AS/400 library list?

Answer If it is an existing file, you may set the *libraryName* attribute in the *AS400File* part's settings to *LIBL. (The special values, *CURLIB, *USRLBL, and *ALL are also acceptable.)

Realize that these special values are interpreted with reference to the host server job, which may be determined by that job's job description. This is not necessarily the same as the user's job description.

If you wish to modify the server job's library list before opening the file, send a #remoteCommand: to do ADDLIBLE, CHGLIBL, and so on. For example:

```
Smalltalk
"created a new AS400File part and now add mylib "

myKeyedFile remoteCommand: 'ADDLIBLE mylib'.
```

You must use an actual library name if you are creating the file on the host server from an existing client record description. If none is specified, it will be created in the library specified by the file part's system userLibrary attribute which is QTEMP by default. That's also where the DDS source is created.

Record Name and DDS

Question When I create a record description and use that record description to create a file, it looks as though the DDS for the file is temporary as generated on the AS400. After the file has been created, the DDS is deleted. Now, is there a way to write the DDS to a file so that it is not deleted?

Answer Yes, it is generated into QTEMP/OFDDSFIL. Since the file is in QTEMP, the file is never explicitly deleted, it is, however, implicitly deleted when the DDM job is ended. The library is specified in E4AS400FileService>>userLibrary (private) so in theory you could change this to be a library name of your choice. Any changes made to IBM code place your support in jeopardy, so I would sugg est the following: If this function is important to you, make a class extension of E4AS400HeteroFileService (V2.0) or E4AS400HomoFileService (V3.0) in one of *your* applications. Add the #userLibrary method to return a string specifying the name of the library. Since E4AS400FileService is abstract, overriding the method in your extension should be safe. As always, this would be overriding private code, so use at your own risk (this is an untested workaround).

Cursor Stability

Question I am reading records from a file with the following settings:

- Relational operator: EQ
- File lock mode: #updaters
- Commit lock level: #cursorStability
- File open: #openReadOnly

I have the feeling that records (keyed file) get locked after a record has been read (*readAt*), even though the file has been opened with *openReadOnly*. Is that possible?

Answer Normally, #openReadOnly does not lock any records. However, since the file is part of a commitment control definition, the rules change slightly. The cursor stability (*CS) commitment control definition locks *all* records. See the *VisualAge for Smalltalk, Version 3 for AS/400 Connection User's Guide*. Under "Accessing AS/400 Data," it says: "Every record accessed for files opened under commitment control is locked."

The difference between *ALL and *CS is that the record that is locked can change when the definition is *CS. If the *ALL definition is used, then record locks accumulate as records are accessed, and not released until a commit or rollback is used.

It is up to you to establish whether or not your application requires the readOnly data to be locked as part of the transaction. If it does, then *CS or *ALL may apply to your case. If not, then do not start commitment control for the read-only file.

Job Queues

Question How can I get AS400JobQueue instances of all job queues that exist on the system (or in my library list)? I want to be able to prompt the user with a list of valid choices.

Answer You can obtain a list of available job queues by using an Object List Part (in the AS/400 Host Object Parts category. In the list settings, change the Type to *JOBQ. You may also want to modify the name and library settings as is appropriate for your application.

Client Access for Windows

Question Is it possible to make Client Access/400 for Windows open automatically when an application is launched?

Answer If you just need to get the router and host server connections started (which is all you need for VisualAge for Smalltalk AS/400 Connection feature), then you could use the *Program Starter* part to launch the CA/400 Advanced Connection dialogue (CAWINEHNECONN.EXE) in your application #startup method. (Do this before doing *AS400System reconfigureAllSystems*.)

This might be more detail than you want to present to your user, but the internal API to get the basic CA/400 sign-on dialog is unavailable to the VisualAge for Smalltalk application developer. Also, there is little that can be done, in terms of return codes or other methods of returning the results of the user′s interaction with the connection dialogue to the VisualAge for Smalltalk application.

You could also use the configured CA/400 Startup, WSTRPCS.EXE, in place of EHNECONN.EXE. In both cases, you may want to use the CA/400 router API to check if the router is already loaded so that you don′t do this unnecessarily.

If your user automatically starts CA/400 at the Windows startup, then this concern can usually be eliminated.

ODBC vs. File Access Part

Question Are there any recommendations or performance benchmarks that would suggest when to use ODBC and when to use the keyed file class to access data on the AS/400?

Answer We have no benchmark comparisons between the AS/400 file parts (DDM) and ODBC that would address all application uses. Experiments that measure response in the kind of file access that is typical of an application are the best way to get reliable data for selecting the righ t choice for your application.

In general, ODBC access will have a certain amount of fixed overhead that makes it prohibitive (compared to DDM) for one-record-at-a-time use. DDM is, up to a point, faster for simple retrieval and update of groups of records. But this is primarily true when there is little need for filtering or selective processing of records in the client application.

If you have more complex queries and updates of groups of records that in DDM access, would require retrieval of many useless records and additional client-side record processing, then ODBC access might show better performance, in spite of its initial overhead. As you saw in your trial, there is some ODBC caching within the client image that cuts out much of the overhead for repetitive record processing.

ODBC also has more alternatives for moving the balance of the processing away from the client and onto the server, such as prepared packages and stored procedures. In DDM, server-side filtering is done by access through logical file views and joins of physical files. The ODBC stored procedure

advantage can be countered on the DDM-side by the use of AS400RemoteProcedureCall parts, although data queues might be more effective with RPCs than DDM file access with RPCs.

As far as programming overhead and maintainability are concerned, again, your application context is the key. The AS/400 Connection feature takes most of the burden out of using DDM. But you may find the additional client-side record processing needs in some parts of your application make ODBC a simpler alternative. Also, if your client application must operate with non-AS/400 servers, ODBC is the only alternative that has a chance of working with both.

Ideally, you will design your application so that you could freely use either DDM, or ODBC, or both, as your application needs dictate. Or, at least be able to switch with a minimum of change as your application evolves. In any case, a small-scaled prototype of the kind that you did will give you the most help in deciding what is best in a specific application context.

File Agents

Question I am using an E4KeyedFile to access data on the AS/400. Before sending an open message, I set the library name to *LIBL. After the connection has been established, how can I determine the name of the library containing the file?

Answer While not officially supported (private and internal methods are utilized), there is a way to determine the real file library after the open in Version 3.0.

First, make sure you are getting the keyed file by asking the broker for an #extendedKeyedFile, not a #keyedFile. Either will answer an E4KeyedFile, but the underlying service will be different.

After performing the open, the file can ask for the real library by invoking the following:

Smalltalk

```
self openFeedbackArea libraryName.
```

Using Windows 16-Bit Platforms over TCP/IP

Tip

Intermittent hang problems occur when you try to access AS/400 information on the Windows 3.1 platform, using TCP/IP for your communication protocol. Do not deploy any applications that would rely on Windows 3.1 TCP/IP and the AS/400 Connection parts. We are continuing to investigate the cause of these problems.

When using any AS/400 Connection parts on Windows 3.1 and TCP/IP as your communication protocol, there is a restriction that communications to the AS/400 can not occur within a Windows 3.1 callback.

Many of the VisualAge parts make use of callbacks. It is not possible to list all situations where a callback can occur from a visual part or to describe when AS/400 parts actually require to communicate to the AS/400 based on actions, events, or attributes being set and requested, because they can be state dependent.

Here are a few of the more common situations you may encounter and some workarounds:

- The *selectedItem* event on a combo box visual part is implemented within a callback. Any connections fired as a result of selecting an item in the list will be run within the callback. If an action or attribute on an AS/400 part is connected to the *selectedItem* event, and that action or attribute requires access to the AS/400, a walkback would occur stating ″Invalid operation during callback.″

  The recommended workaround is to use a drop-down list and entry field rather than a combo box.

- When you use a tree view, the attribute specified in the *itemChildrenAttribues* is called from within a callback. If retrieving the children involves accessing the AS/400, this would fail with a walkback stating ″Invalid operation during callback.″

  The recommended workaround is to use the *itemChildrenRequested* event of the tree view to access the children. This does not use a callback.

- When you use a container view and attributes are set on the columns, the setting of these column attributes occur within a callback. If setting the attributes involves accessing the AS/400, this would fail with a walkback stating ″Invalid operation during callback.″

The recommended workaround is to use a script to retrieve all AS/400 data before setting the items in the container view.

Unspecified Key

Tip If you get the error ″Key xxxx for yyyyyy is not specified″ when attempting to retrieve data for jobs on the AS/400, you have to install the fix provided in the product refresh fixes for the AS/400 Connection.

Chapter 2. Communications and Transactions

In this chapter, we discuss the Communications/Transactions Feature of VisualAge for Smalltalk. Communications protocols supported by this feature include:

- TCP/IP
- APPC
- CPI-C
- NetBIOS
- CICS ECI
- MQSeries
- EHLLAPI

In this chapter, we do **not** discuss VisualAge Web Connection Parts. Refer to Chapter 9, “Web Connection” for more information.

Supported TCP/IP Stacks

Question What TCP/IP stacks are supported and work with the VisualAge Communications Feature for either OS/2, Windows or AIX environment?

Answer The Communications Feature supports the IBM TCP/IP stack on OS/2 and AIX and supports the WinSock Verison 1.1 TCP/IP stack on Windows. Version 3.0a added Windows 95 and Windows NT TCP/IP support.

Testing the TCP/IP Setup

Question How can I test my VisualAge TCP/IP setup?

Answer One thing to check is to inspect the following code in the Transcript:

```
Smalltalk
AbtTCPInetHost localHost
```

Address in Use

Question I have a program that performs a bind on a socket. After closing the socket, the IBM TCP/IP software still thinks the address is in use. Is the problem with the TCP/IP software, IBM Communications, or am I missing something during cleanup?

Answer Actually this is normal operation for TCP/IP. The TCP stack holds that address *in use* even after the socket that was bound to the port has closed. This is done to make sure that the clients who knew about the old program don't try to access the new one. You can set the socket option REUSEADDR to prevent this. But beware, because on some platforms you may end up with two active programs using the same address and there is no way to know which program will service the next client.

Using TCP/IP in Scripts

Question Is there an example of using TCP/IP via Smalltalk scripts?

Answer Using Smalltalk scripts to program TCP/IP is quite a large field. Nevertheless, below is one very simple example.

Server script:

Smalltalk

```
| aCommLink aClass aHost aPort aResult |
aHost := AbtTCPInetHost getHostById: '127.0.0.1'. "loopback"
aPort := AbtTCPPort usingHost: aHost portNumber: 175.
aClass := Smalltalk at: #AbtSocket ifAbsent: [^nil].
aCommLink := (aClass newStreamUsingPort: aPort) socket.
(aResult := aCommLink bind) isAbtError ifTrue:
                    [^aResult display].
aResult := aCommLink listen: 2.
[ aCommLink accept.
  10 timesRepeat: [CgDisplay default bell: 100] rbrk.
  fork.
^aResult.
```

Client script:

Smalltalk

```
| aCommLink aHost aPort |
aHost := AbtTCPInetHost getHostById: '127.0.0.1'. "loopback"
aPort := AbtTCPPort usingHost: aHost portNumber: 175.
aCommLink := aPort newConnectedSocket.
aCommLink sendData: 'hello', (String with: Character cr).
aCommLink soclose.
```

It would be a very good idea to test for errors on the *socket* and *accept* calls. You may also want to look at another piece of code, below, which is interesting in many aspects.

Smalltalk

```
getData
"Public - return text from server"

| abtHost abtPort abtSock rc chunk gData |

abtHost := AbtTCPInetHost getHostByName: self host.
abtHost isCommunicationsError
  ifTrue: [ System errorMessage: 'error host'.
      ^nil ].

abtPort := AbtTCPPort usingHost: abtHost portNumber: self port.
abtPort isCommunicationsError
  ifTrue: [ System errorMessage: 'error port'.
      ^nil ].

abtSock := AbtSocket newStreamUsingPort: abtPort.
self selector isNil ifTrue: [self selector: ''].
abtSock bufferLength: 8192.

rc := abtSock connect.
rc isCommunicationsError ifTrue: [ rc display. ^nil ].
rc := abtSock sendData: self selector.
rc isCommunicationsError ifTrue: [ rc display. ^nil ].
gData := ''.

[ abtSock isConnected ] whileTrue: [
    chunk := abtSock receive.
    chunk isCommunicationsError ifTrue: [ chunk display ].
    gData := gData, chunk contents asString. ].
abtSock disconnect.
```

Socket Program with Time-Out

Question I have been trying to write a client socket program that has a built-in timeout. To do this, I start two processes, one with a delay timer and one that waits for the socket to return some data. I've attached the code below. Can someone please tell me if this is supposed to work? This program works fine if I do not enclose the receive as a separate process.

The following method always reports that the timeout has occurred, even though I know that there was plenty of time for the server to send back the request.

Smalltalk

```
simple: aMessage timeout: seconds

| host address socket result buffer exceededTimeout sem |
host := AbtTCPInetHost getHostById: 'karsloop'.
address := AbtTCPPort usingHost: host portNumber: 2000.
(socket := AbtSocket newStreamUsingPort: address) socket.
result := socket connect.

result isCommunicationsError ifTrue:
      [self error: 'Could not connect'].

socket sendData: aMessage.
sem := Semaphore new.
exceededTimeout := true.

[
   |delay|
delay := Delay forSeconds: seconds.
   delay wait.
   sem signal.
] fork.
[
buffer := socket receive.
   exceededTimeout := false.
   sem signal.
] fork.
sem wait.
exceededTimeout
   ifTrue: [result := 'Exceeded timeout']
   ifFalse:
      [buffer isCommunicationsError
        ifTrue: [self error: 'Could not receive'].
       result := buffer contentsAsString.
      ].
socket disconnect.
^result
```

Answer Use *abtWait* and *abtSignal* instead of *wait* and *signal.*

Sockets and Streams

Question I want to attach a Stream object to a Socket, as this makes it very easy to write programs that deal with Sockets. Is this possible with IBM Smalltalk or VisualAge TCP/IP?

Answer The VisualAge socket support was designed with records, not streams in mind. Currently we don't have any existing code to support what you want to do. However, I think it's really easy and straightforward to write code to manage the stream interfacing with our socket code.

Error: 'A NetBIOS message was ignored'

Question My application is experiencing random errors while communicating via NetBIOS. In test (development system) I get the message: "A NetBIOS message was ignored:" In production I simply get garbage in the receive buffer. The method involved seems to be #ncbComplete: The error seems to happen when multiple sends and multiple receive are performed. I don't fully understand this code:

Smalltalk

```
AbtIBMNetBIOSCall class>>ncbComplete: anInteger
   "This is the NCB completed procecure for the receiver."

    (self netBIOSDictionary includesKey: anInteger)
  ifTrue: [(self netBIOSDictionary at: anInteger) requestComplete.]
  ifFalse: [Transcript cr; show: 'A NetBIOS message was ignored : '.
        Transcript show: (anInteger printString)].
^nil
```

I understood that #ncbComplete is the entry point for a call-in coming when the NetBIOS call is completed and that the call-in is matched with the original request kept in the netBIOSDictionary. In which case is the *ifFalse:* [] branch activated? I wonder why this case is simply ignored and not treated as an error.

Answer #ncbComplete: is indeed the method called by the asynchronous callback into Smalltalk from a C procedure. That C procedure is called by NetBIOS when the asynchronous NetBIOS function completes. What is passed back into Smalltalk is the NCB address. VisualAge keeps the NCB addresses of all asynchronous calls as the keys of the NcbDictionary and associates with it the instance of the NetBIOS object making the call (name, adapter, or session). If a callback is received with an NCB address that is not associated with any instance of a session, adapter, or name (which should never happen), then there is no clear instance to send the error to. Perhaps we could recheck all the objects that are waiting, to see if their NCB completion code has changed and figure out from that who the NCB completion message was meant for. In theory, this error shouldn't happen and the interesting question is did a bad NCB address get sent back by NetBIOS to the C procedure (which is doubtful) or did something happen to the entry in the dictionary.

Accessing COBOL through CICS

Question My application accesses a COBOL program through the *CICS proc dialog*. When I package the application as runtime image, I get a warning message for each field in my commArea, something like: *no implementor of IS_field1* where field1 is one field in my commArea. Everything works fine, but I am not sure why these warning occurs, and how to fix it.

Answer For this case, the packager warnings are acceptable. I suspect that you've built scripts with assistance of the VisualAge script template tool. In your script, you probably have code similar to that shown below:

Smalltalk

```
(self subpartNamed: 'COBOL External function1')
  valueOfAttributeNamed: 'passedVariable1'
  selector: #'IS_passedVariable1'.
```

When saving VisualAge parts, you may have noticed that VisualAge creates interface specification methods for each of the attributes and actions for your part. These methods are named 'IS_<attributeName>'. However, the *APPC proc dialog* part (and several others) construct their interface specification dynamically based on their settings (no 'IS_' methods are created). The VisualAge script template tool currently builds code assuming that all attributes have a corresponding 'IS_' selector.

We may be able to address this annoyance in a future release. For now, don't be alarmed about packager warnings which reference the *dynamic* attributes of your Proc dialog parts.

Primitive Failing When Opening CICS Proc Dialog Settings

Question I wanted to start working with the CICS parts. When I try to open the settings for the CICS Proc Dialog I receive a walkback with the following information: *Primitive failed in: PlatformFunction>>#call:with:with:with: due to OS error 127* I saw in the VisualAge debugger that VisualAge tried to call FAACIC32.DLL <c:int16 'FAACIC32':CICS-EciListSystem> so I checked my CICS installation again, but everything seems okay. I can run CICS applications, the DLL is there and in the LIBPATH, and so on. What is wrong?

Answer You have VisualAge version 3.0. The original Version 3 release did not support running directly on the server. It would run on a client. This is fixed in Version 3.0a. Obtain the refresh.

Handling a Transaction Abend

Question My application will communicate to host system CICS/ESA using VisualAge ECI parts. The application uses the Distributed Program Link (DPL) function. How do I detect and notify the client of transaction abend on the host system? Is it related to the *lastError* attribute?

Answer The CICS ProcDialog part handles errors only one way. It will display a message box with the error. If that is good enough for you, then you don't have to do anything. If you are using the CICS Program and CICS Logical Unit of Work parts, then you have control over how errors are handled. When an error occurs, as during the execution of the ECI call, the *lastError* attribute is filled with an instance of AbtCICSError, which contains the return code (a negative number) and in some cases a four-character abend code. This error attribute contains an attribute *codesAsString* which will return a string that is a little more readable than the return code. For instance, it would return 'EciNoCICS' if the return code was a -1. This error attribute also has a *display* action that will produce a message box just like the one the ProcDialog always shows. The other thing that happens is the *errorOccurred* event is signaled. You can use this error event to do such things as execute the *rollback* action on the CICS Logical Unit of Work, or open a window of your own where you can show some more information

and have Commit and Rollback buttons to allow the user to decide whether to continue or abort the transaction.

CICS ECI and Code Page Translation

Question In the CICS ProcDialog part on the data attributes setting, there is a code page parameter. The Help explains that you have to enter the code page of the remote system (CICS/ESA). Who does the translation from EBCDIC to ASCII and from one code page to another? Is it the conversion table on the CICS/ESA, the CICS client, or VisualAge?

Answer The ProcDialog of the communications parts uses the code page settings for the AbtForeign record that is generated from the parsed header file. The code page setting represents the code page of the target machine (where the transaction will be sent). The CICS communication area data represented by the AbtForeign record will be translated to the target code page by VisualAge before sent out on the network. When the communication area is updated (when CICS returns from the transaction), the data will be in the target machine's code page (the target machine sent the response in its code page). VisualAge will convert the data in the record to the local code page when you send *at: #fieldName* message to the AbtForeignRecord.

CICS also can translate the data. If you wish CICS to translate the data (you need to set up the correct CICS macro exits), then don't specify a different code page on the VisualAge settings page. If you want VisualAge to do the conversion then set the code page and use a transaction id other than CPMI.

MQSeries and VisualAge on AIX

Tip If you want to use VisualAge or IBM Smalltalk on AIX to talk to MQSeries, then you need to have an MQSeries server installed on the AIX box that VisualAge runs on. VisualAge on AIX does not support the MQ client but it does support an MQ server (MQ installed on the local AIX box). If you need to have messages sent to another MQ server on another box, such as OS/2, you need to create transfer queues and channel files as described in the MQ documentation. MQSeries on AIX can be a server by setting up the *inetd* daemon with the proper values as specified in the *MQSeries Distributed* book. I don't believe there is a listener program (runmqlsr) for MQ on AIX as there is on OS/2. You also need to have the latest PTFs for MQ AIX so you don't encounter *melt* problems. There is a MQ fix for handling signals.

MQ: ″No Message Available″

Question I am trying to read MQ messages. I get them successfully when put locally on the same queue. When I send the MQ message to the host and send a reply back after processing, I am unable to see the message in VisualAge, although it can be read by a native REXX program. This is my setup:

VisualAge → local MQM on OS/2 → VisualAge **OK**
VisualAge → local MQM OS/2 → remote MQM MVS
→ PL/I → remote MQM MVS
→ local MQM OS/2 → VisualAge **not OK**

In both cases the create new message is done with *msgid* and *correlid* set to *none*, but probably the interpretation is different. Any hints?

Answer The problem that you encountered was caused by an error in the *toBeLoadedCode* for the pool dictionary AbtMQConstants. MqmiNone was defined in the *toBeLoadedCode* as a blank string rather than an empty string (MqciNone should also be an empty string). As a result, when we create a new message descriptor we set the message ID to blanks. When we use that message descriptor on the get call, it is trying to read only messages from the queue with a message ID of blanks. The fix is to change the *toBeLoadedCode* so that MqmiNone and MqciNone are empty strings (this should be fixed in the refresh). You can fix this another way by changing AbtMQMessage>>defaultDescriptor (instance private method) from:

```
  at: #msgid put: MqmiNone;
  at: #correlid put: MqciNone;
to:
  at: #msgid put: '';
  at: #correlid put: '';
```

The reason you did not see this when you put a message to the queue with VisualAge and then read it with another application using VisualAge is that in both cases you used a blank string as the message ID.

Syncpoint Processing

Question I am trying to do syncpoint processing with MQSeries on OS/2 but have not succeeded. Neither commit nor backout show any reaction whatsoever, meaning that it neither worked correctly nor gave back any kind of error message. I have set the options for syncpoint processing on the queues. What's wrong?

Answer There are a couple of things you need to do in order to use syncpoint processing with VisualAge. First, messages that are read from a queue will be committed or backed out only if the get message options (GMO) are set for syncpoint. The same is true of the puts to the queue. Set the put message options (PMO) for syncpoint. To accomplish this in Smalltalk, use AbtMQqueue>>gmoOptions: and AntMQqueue>>pmoOptions: . Send the value 2 to the methods MqgmoSyncpoint and MqpmoSyncpoint as defined in the AbtMQConstants pool dictionary. Second, we have a bug in the commit and backout methods of AbtMQqueue. The following line in each method should be changed from:

```
    rslt := AbtMQCall commit: self handle.
 or
    rslt := AbtMQCall backOut: self handle.
```

to:

```
    rslt:=AbtMQCall commit: self queueManager handle.
 or
    rslt:=AbtMQCall backOut: self queueManager handle.
```

We were sending the queue's handle not the queue manager handle on the commit and backout, resulting in an AbtError when we tested these methods.

ASCII to EBCDIC Conversion

Question I'm using the MQ classes to do MQ to a /390 host. I do the put with the format string option. When the host receives the message it is not converted to EBCDIC. The channel has CONVERT=YES on it. What am I doing wrong?

Answer Since we already have support in VisualAge for code page conversions, we decided to use that option for MQ conversions. If you want to send a message to a system with a different code page, you can have the data converted by VisualAge before the data is sent across the wire. There are a couple of ways to accomplish this: If you are just sending strings, then you can use the *convertToCodePage:* method of EsString. There are various variations of this method in EsString. If you are sending a more complex structure or integers with different byte orders (little/big endian), then you should use AbtForeignRecord. Set method *codePage:* to the numeric value of the target code page (such as codePage: 37) and method *bigEndian:* to *true* or *false.*

MQI Sample Application

Question Is there a sample application using the VisualAge MQI parts? The Communication/Transaction feature guide does not seem to have one.

Answer Below is a simple script to put a message to one queue and read a message from another queue. This example is for the Version 3 refresh code. While the MQ API has only a few calls, there are many options you can use to create different behavior. What you want to accomplish will determine how to design the visual parts or scripts.

Smalltalk

```
| aConnectionSpec aConnection rc |

"create a connection spec that will define the names
 of the input queue (reply queue), the output queue
 (request queue) and the queue manager"

(aConnectionSpec:=AbtMQSeriesConnectionSpec new)
   userId: 'salkeld';
   requestQueueName: 'VAQMGR.VAQUEUE1';
   replyQueueName: 'VAQMGR.VAQUEUE2';
   queueManagerName: 'VAQMGR'.

"connect to the queue manager and open the queues"

((aConnection:=AbtMQConnection new)
    connectUsing: aConnectionSpec) isAbtError
 ifTrue: [ ^aConnection lastError ].

"put a string to the output queue"

(rc:=aConnection sendString: 'Hello World.') isAbtError
 ifTrue: [ ^aConnection lastError ].

"read a message from the input queue. write the message
 header and data contents to the Transcript"

(rc:=aConnection get) isAbtError
 ifTrue: [ ^aConnection lastError ]
 ifFalse: [ Transcript cr; show: rc descriptor printString; cr; show: rc
contentsAsString ].

"close the queues and disconnect from the queue manager"

(rc:=aConnection disconnect) isAbtError
 ifTrue: [ ^aConnection lastError ].
```

Commit/Rollback with MQ

Question Commit/Rollback are messages sent to a queue. This implies that both commit and rollback are supported at the queue level. The documentation seems to talk about a logical unit of work only. What is the definition of a logical unit of work with respect to MQ? Is it from open to close of the queue? Is it from connect to disconnect to the queue manager? Is it from first get/put to commit/rollback/close?

If it is related to the connect, then does that imply that all queues opened are committed?

Answer The commit and rollback take a queue manager handle rather than the queue handle. All get-put actions to the queue from the last commit or open of the queue until the time you do a commit or rollback will be part of the logical unit of work. All those actions will be committed or backed out as a group. The commit or rollback is on an application or program basis even though the queue manager handle is used. Even if two programs were reading from the same queue and one was processing the messages under a logical unit of work and the other one was not, the messages that are part of the one program's logical unit of work are the only ones affected by the commit or roll back.

Error: MqccFailed

Question I am running the VisualAge 3.0a refresh code, and am running MQ 2.0. When using the visuals, I receive following error message at connection:

```
AbtMQError: rc=2 for #connect in an AbtMQqm at (30.05.96 21.36.26)
→ ('MqccFailed' , 'MqrcQMgrNotAvailable')
```

I found out the the VisualAge MQ parts still looked for the client DLL (I thought this had been improved with some switch) and went into the AbtMQSeriesPMSubapp to change MQIC to MQM:

Smalltalk

```
startUp

PlatformLibrary mapLogicalName: 'MQSERIESDLL'  "$NON-NLS$"
  toPhysicalName: 'MQM'.  "$NON-NLS$"
```

The result is still the same. I can't connect to the queue manager. MQ is installed correctly and the C samples run okay. What could I be doing wrong?

Answer VisualAge defaults to using the client code on OS/2 and Windows 3.1. On Windows 3.1, you have no choice since there is only MQ client code available. On OS/2, if MQ is installed locally, you can go to the subapplication and change MQIC to MQM, swipe the code, and execute it. If you save the method, then the next time you start up the MQM DLL will be used. VisualAge uses the client DLL MQIC since, in most cases, people probably would not want to do the full MQ install on every machine of an MQ user. The client install is much smaller and there is less system maintenance of queue managers, queues, channels and so on.

In your case, if you are using MQM and you not only changed the method but swiped and executed it, then check the case of the queue manager name. MQ is case sensitive and will not find the queue if the case is wrong even if the spelling is correct.

Host Presentation Space

Question What is a host presentation space?

Answer A host presentation space is the 3270 emulator VisualAge is interacting with.

Getting 3270 Cursor Position

Question The 3270 Terminal has an attribute *cursorPosition.* At what time is this attribute updated? I get the cursor position only once when the screen opens. Later on, when the user has moved the cursor around, I never get the new cursor position. Why?

Answer The screen part doesn't update the cursor position, but if you query the position from the screen's terminal, then the updated position is returned (use #cursorPosition).

Intercepting Key Strokes

Question How do I intercept key strokes in a host presentation space for OS/2?

Answer The terminal part has the following methods you can invoke via a Smalltalk script:

startKeystrokeIntercept: takes a string where the first character is the session ID of the presentation space and the second character is a D to intercept AID keystrokes only or an L to intercept all keystrokes. The *getKey:* method takes a record structure *gkeyStruct*. The record structure is defined in the Abt3270HllapiConstants pool dictionary. You must set the *gkeyShortname* field of the record to the session ID from which you are going to read a key stroke. The *getKey:* method will return a zero return code if a key was read and fill in the record structure with the data pertinent to the key you read. If the return code is not zero, then the data in the record structure is not valid. You should read the EHLLAPI programming reference to find how to interpret the data returned in the record. You need to use the *sendKey:* or *type:* methods to put the key you intercepted in the current presentation space (or another presentation space if you wish). You can inform HLLAPI that you accepted or rejected a key stroke you intercepted with the *getKey:* by using *postInterceptStatus:* with *true* (accept) or *false* (reject). After you start key stroke interception with *startKeystrokeIntercept:* you can loop on the *getKey:*, *postInterceptStatus:*, *sendKey:*, or *type:* methods. When you decide to stop intercepting keystrokes, leave the loop and use the *stopKeystrokeIntercept* method.

Accessing COM Ports in Smalltalk

Question What is the correct way to access a COM port from Smalltalk? Are there special classes for that? Or do I have to use C for that?

Answer Although this question is commonly asked as a communications question, it really has nothing to do with the Communications/Transactions feature because you use just base Smalltalk to do that. For an explanation and an example, refer to Chapter 4, “IBM Smalltalk Programming Language.”

Chapter 3. Interface to External Routines

In this chapter, we cover questions and answers related to accessing routines external to VisualAge for Smalltalk.

Using OSSObject, OSObject Pointer, AbtPointer Classes

Question How do I use the OSObject, OSObjectPointer and AbtPointer classes?

Answer Below are a few ideas for using OSObject subclasses. The OSObject hierarchy replaces the V2 OSStruct/OSPointer hierarchies:

Smalltalk

```
" Create an OSObject which points to OS memory "
| x y pf |

 pf := PlatformFunction ...

 " Allocates operating system memory large enough to
 store a MyOSObjectClass instance "

 x := MyOSObjectClass calloc.

 " Creates an instance of MYOSObjectClass. The data for this
 instance resides in a ByteArray in Smalltalk memory"

 y := MyOSObjectClass new.

 " Code below passes the integer address of
 the OSObject to the function "

 pf callWith: x abtAsExternalPassedPointer.
 pf callWith: y abtAsExternalPassedPointer.

 " You can create accessor methods in MyOSObjectClass
 to store/extract the data values. VisualAge provides
 a tool that automatically builds OSObject subclasses."

 x inspect.
 y inspect.
```

In the code above, x and y can be treated equivalently in your code. OSObject instances function in the same manner whether they contain the actual data (such as y) or a pointer to the data (such as x). The methods *abtMoveToOSMemory* and *abtMoveFromOSMemory* can be used to toggle between storing data as an address to external memory or a ByteArray in Smalltalk memory. Be careful: If *abtMoveToOSMemory* is used to create a pointer, the pointer will be freed when the OSObject subclass is garbage collected. #abtMoveFromOSMemory copies the structure to Smalltalk memory and frees the pointer.

Smalltalk

```
MyOSObjectClass new abtMoveToOSMemory.
```

Smalltalk

```
MyOSObjectClass calloc abtMoveFromOSMemory.
```

AbtPointer probably won't be of much use to you. This class adds a VisualAge interface specification to OSObjects to simplify pointer manipulation for simple datatypes (integer, character) when using the C External Function part.

Smalltalk

```
| pointer |

pointer := ( AbtPointer calloc: 4 )  pointerDescriptor:
  ( AbtCPointerField new pointerType: AbtCIntField new ).
pointer valueAtAddress: 12.
pointer valueAtAddress inspect.
pointer free.
```

OSObjectPointers are used to store pointers to other OSObject subclasses. I have not made much use of this class, but it might be useful if you need to manipulate an array of pointers:

```
Smalltalk
| osObjectPointer arraySize |

osObjectPointer := OSObjectPointer itemType: MyOSObject.
arraySize := 1.
" Allocate enough space to store 3 pointers "
osObjectPointer reference: ( ByteArray new: (arraySize * 4 )).
osObjectPointer at: 0 put: MyOSObject calloc.
( osObjectPointer at: 0 ) inspect.
0 to: arraySize -1 do: [ :i |
 ( osObjectPointer at: i )  free ].
```

Freeing a DLL

Question I have tested a DLL function in a VisualAge for Smalltalk application and now I cannot replace the DLL file. It's in use. How can I get VisualAge to turn it loose?

Answer I assume that you have an application where you dropped the COBOL external function part and set it up to point to some DLL. After some testing, you decided you wanted to replace the DLL with an updated version. You brought down the application and the composition editor window so the COBOL external function part would go away. But, the DLL is still open, so you can't replace it. Here is a simple way. After you make sure all the windows that might reference your DLL are gone, execute the following from the Transcript:

```
Smalltalk
(PlatformLibrary logicalName: 'yourdll') close
```

Error: Abt.154e

Question I am getting the following error: Abt.154e. What could be the cause?

Answer Some common causes for Abt.154e are:

- DLL does not exist in the LIBPATH (OS/2) or PATH (Windows).
- A DLL that is invoked from the called DLL cannot be found in the PATH or LIBPATH. That is, VisualAge for Smalltalk calls A; A.DLL calls B.DLL (both A and B must be in the LIBPATH concatenation).

- The entry point name was entered incorrectly. Be sure that the entry point name is keyed in exactly as it appears in the ′.def′ file. Depending upon the compile or link options specified, the entry point name may be case-sensitive (ABC differs from abc).

Signaling the End of a Rexx Program

Question How can I signal to VisualAge for Smalltalk the end of a Rexx program started with program starter?

Answer If you are interested in being notified when a program started from the program starter ends, the best way to proceed is to subclass the AbtProgramStarter. This part uses the DosStartSession API to start your programs. DosStartSession can be invoked so that the OS/2 session manager will write a data element into a queue that you specify when any child session ends. Your subclass needs to create the queue and issue the DosReadQueue to be notified when the child session ends. See the OS/2 CP Reference for more information on this API.
AbtProgramStarter>>startProgram executes the DosStartSession.

I believe that you can also retrieve the return code of your program with this technique.

Set Working Directory for Program Starter

Question How can I set the working directory for a program started from the program starter? When I launch an application using the program starter, the working directory seems to be set to the root of the system drive, which is not the one I want.

Answer The best way to do this is to create a subclass of the program starter part to provide a method for setting the working directory before launching the program starter. Here′s the code that will do it:

Smalltalk

```
AbtProgramStarter subclass: #ProgramStarterSubclass
  instanceVariableNames: 'dir '
  classVariableNames: ''
  poolDictionaries: ''
```

The public methods for this class would be similar to the following:

```
Smalltalk
dir: pathName
"Set the directory to pathName"

(CfsDirectoryDescriptor chdir: pathName) isCfsError
ifTrue:
  [Transcript cr; show: 'Fail'.
  ^nil].
dir := pathName.
self signalEvent: #dirChanged with: pathName.
```

```
Smalltalk
dir
"Answer the current directory to the receiver"

dir isNil ifTrue:
  [self dir: CfsDirectoryDescriptor startUpDirectory].
^dir.
```

```
Smalltalk
changeDir
"Change directory"

| directory |
directory := self dir.
directory isNil
ifTrue: [^nil].
(CfsDirectoryDescriptor chdir: directory) isCfsError
ifTrue:
  [Transcript cr; show: 'Fail'.
  ^nil].
```

Don't forget to do an *editPart* on your subclass so you can promote *dir* and *changeDir* to the public interface.

Debugging DLLs

Question Is there a way to debug a C-DLL which is called by VisualAge in the development environment? Is there a way to invoke the IBM Presentation Manager Debugger?

Answer You load VisualAge under the PM debugger and set a DLL-load breakpoint. Once the DLL is loaded, you can set source breakpoints. If you get a trap in the DLL, IBM Presentation Manager Debugger will show the offending line.

You should add exception handling to your code. Attached you find a simple example that can be adapted for use in a DLL. By the way, in the refresh an exception in a primitive will return back to Smalltalk instead of letting the operating system handle it.

c

```
#define INCL_DOS
#include <os2.h>
#include <setjmp.h>
#include <stdio.h>
#include <string.h>

extern int main(void);

/* Exception registration record.  Stored on stack, with first pointer
 * to next registeration record, second pointer to exception handler,
 * and the rest defined by the author of the exception handler. */

typedef struct
{
  struct _EXCEPTIONREGISTRATIONRECORD * volatile prev_structure;
   _ERR * volatile ExceptionHandler;
   jmp_buf env;
} MYEXCEPTIONREGISTRATIONRECORD,
          *PMYEXCEPTIONREGISTRATIONRECORD;

/* Exception handler that returns traps via longjmp(). */

extern ULONG APIENTRY MyExceptionHandler
    (PEXCEPTIONREPORTRECORD pReportRecord,
    PEXCEPTIONREGISTRATIONRECORD pRegRecord,
    PCONTEXTRECORD pContextRecord, PVOID pReserved)
```

C (continued)

```
 {
  ULONG rc = XCPT_CONTINUE_SEARCH;
  if (pReportRecord->ExceptionNum == XCPT_ACCESS_VIOLATION)
    longjmp(((PMYEXCEPTIONREGISTRATIONRECORD)
pRegRecord)->env, -1);

  /* If we return to here then we could not handle the exception. */

  return rc;
 }

 extern BOOL Trapper(PSZ psz)
 {
  MYEXCEPTIONREGISTRATIONRECORD myExceptionRegRecord;

  /* Insert my exception handler into the chain. */

  myExceptionRegRecord.prev_structure = NULL;
  myExceptionRegRecord.ExceptionHandler = MyExceptionHandler;
  DosSetExceptionHandler((PEXCEPTIONREGISTRATIONRECORD)
    &myExceptionRegRecord);
  if (setjmp(myExceptionRegRecord.env))
    goto OnException;

  /* Now go about my business in safety. */

  if (strlen(psz))
    printf("Trapper says okay to '%s'\n", psz);
  else
    printf("Trapper says it is empty\n");

  /* I'm done, so unchain my exception handler. */

 DosUnsetExceptionHandler((PEXCEPTIONREGISTRATIONRECORD)
  &myExceptionRegRecord);
  return TRUE;
```

C (continued)

```
  /* The code below is only executed if a trap occurs. */

OnException:
  printf("Trapper says 'ouch!'\n");
DosUnsetExceptionHandler((PEXCEPTIONREGISTRATIONRECORD)
  &myExceptionRegRecord);
  return FALSE;
}

extern int main()
{
  Trapper("Hello");
  Trapper(NULL);
  Trapper("");
  Trapper((PSZ) 42);
  Trapper("Goodbye");
  return 0;
}
```

Calling OS/2 Presentation Manager API Functions

Question Is there a way to call OS/2 Presentation Manager API functions?

Answer Yes, there are methods for calling Win... functions. Let's illustrate this by an example.

c

```
BOOL WinFillRect(HPS hps, PRECTL prcl, long lColor);
```

This function operates on an HPS and therefore the method belongs to the class OSHps. All other arguments to the function are passed as message arguments; thus, the method is

```
OSHps>>winFillRect:lColor:
```

If you look at the implementation of the method, you can see that there is WinFillRect pointing to a PlatformFunction defined like

```
<c: bool 'PMWIN':743 uint32 pointer int32>
```

Function Like f(int*)

Question I want to pass an integer to a C DLL in such a way that it can modify it and, after the function returns, I can read the modified value. What do I do?

To do this, is it a good idea to pass a pointer to the integer to a PlatformFunction? If yes, what's the proper way to do so? The handy message #asPointer seems to be classified as Obsolete, and I can't read the contents reliably anyway. Having to create a OSBasicType and then an OSObjectPointer seems like overkill.

Answer You might consider using a ByteArray or an OSObject instance to pass your pointer to integer. Define the parameter to be passed as a #pointer in your PlatformFunction definition. When a ByteArray is passed as an argument to a PlatformFunction expecting a #pointer argument, VisualAge for Smalltalk passes a pointer to that ByteArray to the external function. Try this:

Smalltalk

```
| pf intPointerParm result |

pf := PlatformFunction
callingConvention: 'c'
function: 'test'
library: 'test'
parameterTypes: #(pointer)
returnType: #void.

intPointerParm := ByteArray new: 4.

" Put an input value into the ByteArray "
intPointerParm int32At: 0 put: 12.

" call the function  ->  pf callWith: intPointerParm "

" pf callWith: intPointerParm. "
result := intPointerParm int32At: 0.
```

If you need to retain the address of your pointer after the function call, you must use an OSObject to pass the parameter. Your VisualAge for Smalltalk application must explicitly free the allocated memory when you are finished using the address. Use this:

Smalltalk

```
| pf intPointerParm result |
pf := PlatformFunction
 callingConvention: 'c'
 function: 'test'
 library: 'test'
 parameterTypes: #(pointer)
 returnType: #void.

intPointerParm := OSObject calloc: 4.

" Put an input value into the OSObject " intPointerParm int32At: 0 put: 12.

" call the function  ->  pf callWith: intPointerParm
abtAsExternalPassedPointer "

" pf callWith: intPointerParm. " result := intPointerParm int32At: 0.

intPointerParm free.
result inspect
```

Legacy Code

Question How is legacy code handled?

Answer Legacy COBOL, C, or C++ applications may be wrappered into parts using the COBOL, C, and C++ support included with the base product. The IMS Connection feature provides the ability to wrap existing IMS transactions to be used as parts within the VisualAge for Smalltalk environment. IBM SOM and DSOM support will allow you to reuse existing SOM and DSOM objects as a VisualAge for Smalltalk part.

Calling a Smalltalk Image from OS/2

Question How do I call the VisualAge for Smalltalk image from a separate running program under OS/2?

Answer One way is to use the nonblocking API support to wait on a semaphore or a named pipe. Look also in the *IBM Smalltalk Programmer's Reference*, Version 3.0, Chapter 11.There is an interesting function BOOLEAN EsPostAsyncMessage(EsVMContext vmContext, EsObject receiver, Object selector, U_32 argumentCount, ...) that can be called from a thread in a DLL. Yes, the DLL must be loaded. The EsPostAsyncMessage call must be made in a thread in the Smalltalk process. So there is some initialization that you must do. The DLL must be written to handle whatever IPC mechanisms you want to use.

Calling Smalltalk from the Outside

Question I have a Smalltalk event handler that needs to receive messages from an outside system. I need to find out how an external application such as a C program can talk or execute a Smalltalk method. In C, I would just create a DLL with the necessary functions, but I can find no Smalltalk equivalent. Chapter 8, "IBM Smalltalk Virtual Machine API," of the *IBM Smalltalk Programmer's Reference* mentions "calling IBM Smalltalk from other languages" in the introduction, but doesn't discuss it in detail anywhere.

Answer Listed below is a simple example of a user primitive written in C that calls into the Smalltalk interpreter by sending a message. There are two functions in the C program. The first function is called by a Smalltalk method and passes the receiver and selector to be used by the second C function to send messages back into Smalltalk. The second C function runs on a new thread created by the first function. The second function performs a loop that results in sending four messages back into Smalltalk. The Smalltalk method that receives the message from the C function updates a List with the string "Wake Forest" whenever it is called.

Hopefully this simple example, along with what's in the *IBM Smalltalk Programmer's Guide*, will get you started:

Smalltalk

```
setUp
"Sends a message to initUpdateListRoutine passing a receiver
(an instance of UserPrimitiveExample2) and a selector (updateList).
The initUpdateListRoutine will call the user primitive C function with the
receiver/selector parameters.  The C function can then use this
information to
call back into Smalltalk."

self initUpdateListRoutine: self makeFixed
 selector: #updateList makeFixed.
```

Smalltalk

```
initUpdateListRoutine: receiver selector: selector
"Two arguments are actually passed to the C routine.  The arguments
passed are the same ones received by this method. "

<primitive: 'samprim2' :initUpdateListProc>
^self primitiveFailed
```

Smalltalk

```
updateList
" This is the method that is called by the C function. "

| col |
[ col := (self subpartNamed: 'List') items.
  col add: 'Wake Forest'.
  (self subpartNamed: 'List') items: col.
] abtDefer.
```

Then use this:

c

```
/* User primitive example */

 #define INCL_DOSPROCESS   /* Process and thread values */
 #include "esuser.h"
 #include <os2.h>
 #include <stdio.h>

/* Now for the globals */
 static ESGlobalInfo * globalInfo;
 static EsObject receiver;
 static EsObject selector;

 void getListData(void);

/* This routine gets the Object and Message selector needed */
/* for calling the Smalltalk routine. */

 EsUserPrimitive(initUpdateListProc)
 {
   EsObject result;
     TID  ThreadID;     /* New thread ID (returned) */
     PFNTHREAD  ThreadAddr;     /* Program address */
     ULONG  ThreadArg;     /* Parameter to thread routine */
     ULONG  ThreadFlags;     /* At start thread, how to allocate stack */
     ULONG  StackSize;     /* Size in bytes of new thread's stack */
     APIRET  rc;     /* Return code */

     ThreadFlags = 0;     /* Thread is to be started immediately */
     StackSize = 4096;     /* Set the size for the new thread's stack */

   globalInfo = EsPrimVMContext->globalInfo;
   receiver = EsPrimArgument(1);
   selector = EsPrimArgument(2);
   rc = DosCreateThread(&ThreadID, (PFNTHREAD)getListData,
     0L, ThreadFlags, StackSize);
   EsPrimSucceed(0);
 }
```

C (continued)

```
void getListData ()
{
  EsObject result;
  int I;

  for (i = 0; i < 4; I++)
  {
    DosSleep(1000);
    EsSendMessage(globalInfo->currentVMContext, &result,
      receiver, selector, 0);
  }
  return;
}

EsDefinePrimitiveTable(samplePrimitiveTable)
EsPrimitiveTableEntry("initUpdateListProc", initUpdateListProc)
  EsEndPrimitiveTable
```

Interface to Native Presentation Manager Widgets

Question How do I use existing Presentation Manager controls in Smalltalk? That is, I want to register my own PM window procedure, and so on.

Answer Wrapping a PM control requires considerable knowledge of PM and how Smalltalk OSWidgets work. In the past, only independent software vendors added new OSWidgets. Now that Version 3 includes the source code needed to wrap a PM control, you could do it but I suggest finding an easier approach since it is undocumented and requires private APIs; the methods change from release to release.

Nevertheless, a very interesting sample of how you can do it is included on the product CD. The filename is DIALOG.DAT. Import it to your library, then load it to your image, and enjoy!

Changes to 16-Bit Function Calling Convention

Question I'm calling a 16-bit DLL as a PlatformFunction. It works fine using VisualAge for Smalltalk Version 2.0, but fails in the DLL with a system trap using VisualAge for Smalltalk Version 3.0. What am I doing wrong?

Answer Two changes to PlatformFunction have occured for Version 3, both of which are involved in the problem.

- The name of the *pascal 16* calling convention has been changed. *Pascal* which was the 16-bit convention is now the 32-bit convention. So if you are calling a 16-bit DLL, use the *pascal16* calling convention. Inspect the following code snippet for a complete list of valid calling conventions:

  Smalltalk

  ```
  PlatformFunction callingConventions
  ```

  Also, consult p. 352 of the *VisualAge for Smalltalk User's Guide* for additional information on calling conventions.

- The second problem is that OSObject's behavior has changed.

This is all a little disconcerting considering that both of these object's behavior are listed as API.

Calling OSObjects

Tip One thing to be aware of when using the C External Function part is that char* fields are stored as actual 4-byte addresses to OS memory. Fields of this type can be set in several different ways:

1. Set the field to an OSObject instance (a pointer). VisualAge for Smalltalk stores the address of the OSObject into the structure. The programmer must manage the memory.
2. Set the field to a String. VisualAge for Smalltalk allocates OS memory large enough to contain the string, then copies the contents of the string to the allocated memory.

Caution

If you change the contents of the String, VisualAge for Smalltalk copies the new String to the address already saved in the structure. You should initialize fields with type 'char *' to contain the largest expected String. VisualAge frees the allocated memory when the part is destroyed.

Memory Leaks from a C DLL under OS/2

Question I am calling a C DLL which allocates considerable storage and then frees the storage when it is done. The OS/2 swap file seems to grow without bound even though the storage is freed. What's wrong?

Answer I would suggest that you get the Theseus tool available from IBM and use its memory-leak detection option. It is a little tedious matching the allocates and frees, but you should be able to determine if your application is increasing its memory usage and which DLL is allocating the memory that is not being freed.

Debugging C DLLs Called from VisualAge

Question Is there a debugger that will allow me to debug a C DLL that was called from VisualAge for Smalltalk?

Answer Yes. On OS/2 I use IPMD. The key thing is to first set a LOAD breakpoint for your DLL so that when it gets loaded because you tried to execute a function In it, then you will cause a break that will allow you to set break points in the DLL itself.

Checking If Platform Function Is Available

Question I am calling a C DLL through a Platform Function. This works fine. But now I want to check if the DLL is really available, before making a call to it.

How can I ask the Platform Function (or Platform Library) if it is available (its DLL is available)?

Answer Checking if the DLL is available before actually calling it sounds a good idea because if you call the DLL via PlatformFunction when it is not available, the VisualAge for Smalltalk image crashes.

One way to fix this problem is to handle an exception by sending #when:do: to a block that tries to access your platform function. If there is any problem, then the block at the second parameter gets executed. There you can send #primitiveErrorCode to your platform function to get the OS error (2 is for DLL not found, 127 is procedure not found, and so on) and do whatever you want including resuming execution.

There is also a direct way to see if the function is available. The PlatformFunction>>#abtAddress is useful for seeing if a DLL can be opened. The #abtAddress routine answers the function address or an AbtError object containing the operating system return code. Use this:

Smalltalk

```
" Inspect the output of this code for a function that exists and a function
that does not exist. "

| func1 |
func1 := PlatformFunction callingConvention: 'c'
 function: 'func1'
 library: 'mydll'
 parameterTypes: #(pointer)
 returnType: #none.
func1 abtAddress
```

#asPointer Method

Question I have noticed that the *asPointer* method is obsolete. What is now the way to put a String as a char * member of a structure (subclass of created as a record wrapper)?

Answer The method *copyToOSMemory* is a simple way to create a pointer to a String. This method creates a new instance of the class OSStringZ (an OSObject subclass), and copies the String to the memory occupied by the OSStringZ. I notice that this method is defined in application WindowsAndPMPlatformFramework, so I'm not certain about its existence or implementation on other platforms.

Smalltalk

```
'aaaaaaaaaaaaa' copyToOSMemory.
```

COBOL Wrapper Locking the DLL

Question I am using a COBOL validation subroutine which is called from VisualAge for Smalltalk using the COBOL part. VisualAge for Smalltalk appears to lock the DLL once the part has run, so if I want to refresh the COBOL DLL after recompiling and linking, I am prevented by a system message saying the DLL is in use by another process.

Shutting VisualAge for Smalltalk down doesn't help either as saving the image after running the part, locks the whole machine up and we have to resort to switching off and on!

Answer The COBOL part has an action *closeLibrary,* which can be used to close your dynamic link library. This action is useful at edit time for cases like yours where you would like to replace the DLL. To see how this works, add a temporary push button to your view and connect the *clicked* event to the *closeLibrary* action of the COBOL part.

Note:

You must click **More...** from the COBOL part in order to see the *closeLibrary* action.

Caution:

This works for **most** DLLs, but we have encountered cases where it does not (causes a walkback).

Passing a Complex Structure to a C DLL

Question How do I pass a complex structure as a parameter to a C DLL? That is, a structure that contains multiple sub-structures? I've created an AbtCompoundType for each substructure, but I don't know how to package everything together in one memory area.

For example, I used this:

```c
typedef struct
{
  char fieldName[32]
  char fieldValue[32]
} FIELDS;

typedef struct
{
  char  abc[32];
  char  bcd[32];
  char  def[32];
  FIELDS fieldpairs[32];    ″ an array of fields ″
} POLDETAIL;

typedef struct
{
  char  asd[32];
  char  sdf[32];
  FIELDS fieldpairs[32];  ″ another array of fields ″
} COVDETAIL;

typedef struct
{
  char  resultName[32];
  char  resultValue1[32];
  char  resultValue2[32];
} RESULTS;
```

Then my function prototype was as follows:

```c
POLDETAIL    *pPol1;
COVDETAIL    *cCov1;
RESULTS      **pResults;

void CALC(POLDETAIL *pPol1,
        COVDETAIL *pCov1,
        RESULTS  **pResults )
```

Answer You can create a Compound type that has embedded Compound types within it (substructures within structures). An example from MQ record structures:

Smalltalk

```
| rec |

rec:= AbtCompoundType new name: 'QMQueueAliasInquire';
  addField: ((AbtCompoundType new name: 'QMQueueInteger';
  addField: (AbtCLongField new name: #MqiaDefPersistence; yourself);
  addField: (AbtCLongField new name: #MqiaDefPriority; yourself);
  addField: (AbtCLongField new name: #MqiaInhibitPut; yourself);
  addField: (AbtCLongField new name: #MqiaQType; yourself);
  addField: (AbtCLongField new name: #MqiaInhibitGet; yourself);
  addField: (AbtCLongField new name: #MqiaScope; yourself); yourself)
     new name: #QMQueueInteger; yourself);
  addField: ((AbtCompoundType new name: 'QMQueueCharacter';
  addField: (AbtCCharArrayField new name: #MqcaQName; count: 48;
     yourself);
  addField: (AbtCCharArrayField new name: #MqcaQDesc; count: 64;
     yourself);
  addField: (AbtCCharArrayField new name: #MqcaBaseQName; count: 48;
     yourself); yourself)
  new name: #QMQueueCharacter; yourself);
  yourself.
rec := rec newRecord.
```

The variable record now is a record with multiple substructures within it. To pass it on a platform, call you need to use abtAsExternalPassedPointer and the platform call must have that parameter defined as a pointer. In your case, the platform call may look something like this:

Smalltalk

```
| rec1 rec2 rec3 temp1 |
rec1:= (AbtCompoundType ...) newRecord.
rec1 at: ____ put: ___.
rec2:= (AbtCompoundType ...) newRecord.
rec2 at: ____ put: ___.
rec3:= (AbtCompoundType ...) newRecord.
rec3 at: ____ put: ___.
(temp1 := AbtHeapObject calloc: 4) uint32At: 0 put: rec3
abtAsExternalPassedPointer.
(PlatformFunction callingConvention: 'abtsystem'
    function: 'CALC'
    library: 'MYDLL'
    parameterTypes: #( #pointer  #pointer #pointer)
    returnType: #none)
coroutineCallWithArray:  ((Array new: 3)
  at: 1 put: rec1 abtAsExternalPassedPointer;
  at: 2 put: rec2 abtAsExternalPassedPointer;
  at: 3 put: temp1 abtAsExternalPassedPointer;
  yourself).
temp1 free.
```

Notice that since you pass a pointer to a pointer for the result, we got some OS storage and stored the pointer of the record structure in it. We then pass the pointer of that data area in the platform call. On return, remember to free that data area. The results will be in the result structure.

Parsing COBOL COPY-Book

Question When I parse a COBOL COPY-book, I get only 01-level fields. How could I access the lower level fields, such as 02, 05?

Answer If you are using the COBOL External Function part from the Composition Editor, you use the tear-off attribute option to expose the subattributes of your record.

For example, if you would like to work with the 'FIRST-NAME' field of the record below, you would:

- Parse the copybook below from the COBOL external function part
- Save the settings for the part
- Tear off attribute *atmSampleCusts*

- Tear off attribute *cust1* from the previous torn-off attribute
- The attributes for *cust1* will contain the simple (level-03) data items:

COBOL

```
01 ATM-SAMPLE-CUSTS.
   02 CUST-1.
      03 FIRST-NAME          PIC X(20).
      03 LAST-NAME           PIC X(20).
      03 PIN-NUMBER          PIC X(5).
      03 CHECKING-BALANCE    PIC 9(7)V99.
      03 SAVINGS-BALANCE     PIC 9(7)V99.
```

The Smalltalk code below shows how you would manipulate the data structure using Smalltalk code:

Smalltalk

```
| record recordShape level01s cust1 firstName |

" Parse the file and build dataStructures dictionary "
level01s := ( AbtCOBOLLangParser parseFile: 'atm.cpy' )
 dataStructures.

" Extract the definition for the level-01 item "
recordShape := ( level01s at: 'ATM-SAMPLE-CUSTS'
   asSmalltalkGlobalIdentifier ).

" Build an AbtRecord based on the level-01 item "
record := recordShape newRecord.

" Extract the level-02 item (a subrecord)  "
cust1 := record at: 'cust1'.

" Extract the level-03 item 'FIRST_NAME' "
firstName := cust1 at: 'firstName'
```

Calling C Functions from VisualAge on AIX

Question I am working on a proof of concept and need to call C routine called TripsRoute() in a library called trips.a. To do this quickly, I dropped a C external function part, parsed a header file successfully, and used quick-form to lay out a window with some input parameters. On the DLL page I specified trips as the DLL and TripsRoute as the entry point name. I verified that trips.w is in my LIBPATH, but when I execute the function I get an error message:

> ABT.ABT.154.e: Unable to find function 'TripsRoute' or module 'trips'. Operating system return code is 1.

I built trips.w with the table as directed in the manual (*How to create a DLL in AIX*), adding the user primitive table that was created by VisualAge for Smalltalk

Answer When calling a function on the AIX platform, you must define a function mapping table to map the entry point called by IBM Smalltalk to the actual entry point in your library. This allows IBM Smalltalk to access multiple entrypoints within the same library without having to relink the Smalltalk Virtual Machine.

Below is a sample C file used for building an AIX module that contains the function mapping support described above.

The *VisualAge for Smalltalk User's Guide* contains information similar to what is shown below:

c

```
#include <esuser.h>        /* located on your VA server */
#include <myheader.h>        /* myheader is the header file
                            that contains the prototype for
                            your function. 'entryPoint' in this
                            example  */

/* The quoted "entryPoint" is the function name that you specify
   in your VisualAge platform function
   This name does NOT have to match the actual function name */
EsDefinePrimitiveTable(MyFunctionsTable)
EsPrimitiveTableEntry ("entryPoint", entryPoint)
EsEndPrimitiveTable
```

Use the following:

Make File

```
MODULE_NAME = mylib          #the 'DLL'
TABLE_NAME =  MyFunctionsTable  # the single entry point for the lib.
USER_OBJS  =  mylib.o otherlib.o

# the location of 'esuser.h'
SYSTEM_INCLUDE = -I/usr/visualage/..
USER_INCLUDE = -I.

LDFLAGS = -H512 -T512
CFLAGS  = -O -s $(SYSTEM_INCLUDE) $(USER_INCLUDE)
# Specify the libraries that should be linked into your mapping table
LIBS = -lc  -lotherLibs

$(MODULE_NAME): $(USER_OBJS)
  ld -o $@ $(USER_OBJS) $(LDFLAGS) $(LIBS) -e$(TABLE_NAME)
```

Sample Callback Function

Tip

Below is a simple example of a callback function that calls back into Smalltalk when a CICS ECI call has finished. From the example you can fashion your own callbacks. This is also covered in the *VisualAge for Smalltalk Programmer's Guide*.

This is the function:

c

```
#include <esuser.h>
#include <cics_eci.h>

#ifdef OS2
  #define CALLINGCONV _System
#else
  #define CALLINGCONV
#endif

/*****************************************************************************/
/*  Now for the globals                                                      */
/*****************************************************************************/

static   ESGlobalInfo * CicsGInfo;
static   EsObject       CicsReceiver;
static   EsObject       CicsSelector;

/*****************************************************************************/
/*  Return the notification that CICS finished the transaction       */
/*****************************************************************************/
void CALLINGCONV CICSPost(short messageQualifer)
{
  U_32    returnCode;
  EsObject msgQ;

  EsDefineUserPrimitiveEnvironment(CicsGInfo);
  returnCode = EsU32ToInteger((U_32)messageQualifer, &msgQ);
  if (returnCode == EsPrimErrNoError) {
    EsPostAsyncMessage( CicsGInfo->currentVMContext,
                  CicsReceiver,
                  CicsSelector,
                  1,
                  msgQ);
  }
  return;
}
```

C (continued)

```
/*****************************************************************************/
/* This routine starts the Async-notification post routine.  It needs */
/* an Object and Message selector to be used for the call back.      */
/*****************************************************************************/

EsUserPrimitive(getCICSPostRoutine)
{
  U_32    returnCode;
  EsObject postAddress;

  if (EsPrimArgumentCount |= 2) {
    EsPrimFail( EsPrimErrInvalidArgCount, EsPrimArgNumNoArg);
  } /* endif */

  CicsGInfo = EsPrimVMContext->globalInfo;
  CicsReceiver = EsPrimArgument(1);
  CicsSelector = EsPrimArgument(2);

  returnCode = EsU32ToInteger( (unsigned long)CICSPost,
&postAddress);
  if (returnCode == EsPrimErrNoError) {
    EsPrimSucceed( postAddress );
  } else {
    EsPrimFail(returnCode, EsPrimArgNumNoArg);
  } /* endif */
}

#ifdef LINKED_USER_PRIMITIVES

EsDefinePrimitiveTable(AbtCicsPrimitiveTable)
  EsPrimitiveTableEntry("getCICSPostRoutine", getCICSPostRoutine)
#ifdef RS6000
  EsPrimitiveTableEntry("CicsExternalCall", CICS_ExternalCall)
  EsPrimitiveTableEntry("CicsEciListSystems", CICS_EciListSystems)
#endif
EsEndPrimitiveTable

#endif
```

In the above C example, there are two procedures:

1. getCICSPostRoutine will be called from Smalltalk to get the address of the callback function and to set the receiver and selector (method) that should be called on an asynchronous callback into Smalltalk.
2. CICSPost will be the procedure called by CICS when the ECI call has finished. The messageQualifier is passed by CICS in the call to this procedure. CICSPost will convert that parameter into an EsObject and pass it back to Smalltalk in the callback.

Look at AbtCICSLuw class private methods. postRoutineAddress method will call a primitive that calls the getCICSPostRoutine C procedure to obtain the callback address. It also does the important task of setting AbtCICSLuw in fixed storage so it will not be moved when garbage collection occurs. It does this with the makeFixed method. It passes both itself as the receiver and the *requestCompleteFor:* as the method to be used in the callback. The *requestCompleteFor:* method. when called in the callback, will look up in a dictionary the messageQualifier key and find the associated object. We add the key and associated object to the dictionary before an ECI call is made.

The Smalltalk code is this:

Smalltalk.

```
postRoutineAddress

 PostRoutineAddress == nil ifTrue: [
  PostRoutineAddress := self getPostRoutineFor: self makeFixed
      usingSelector: #requestCompleteFor: makeFixed].
 ^PostRoutineAddress
```

Smalltalk

```
getPostRoutineFor: object usingSelector: aSelector

<primitive: 'Abt_CICS_Primitives':getCICSPostRoutine>

^self primitiveFailed
```

Smalltalk

```
requestCompleteFor: qualifier

 ″Post the required semaphore for this qualifier″

 | transaction dictionary |
 AbtTrace point: ′Receipt of message from CICS OS/2′  ″$NON-NLS$″
  withData: qualifier.
 transaction := (dictionary := self luwDictionary)
   keyAtIndex: qualifier ifAbsent: [^nil].
 (dictionary atIndex: qualifier)
  dateAndTime: (Date dateAndTimeNow);
  origin: transaction;
  locus: #requestCompleteFor:withRc:withAbend: .
 transaction semaphore abtSignal.
```

DDE on AIX

Question Is DDE supported on AIX?

Answer DDE is not supported in VisualAge for Smalltalk for AIX, only on OS/2 and Windows.

Chapter 4. CICS and IMS Connection

In this chapter, we cover the CICS and IMS Connection feature.

Successor Uniqueness Violation Exception

Question I get a *successor uniqueness violation* exception, and I don't know what the reason is. It seems like it could be several things, but how do I tell which it is?

Answer Here is a list of the most common reasons for this exception. In this context, *successor* means a Transaction Record that succeeds the one currently processing data received from the host.

1. The transaction sends an unknown screen.

   This occurs if either the screen is not modeled, or there is no transition defined to reach it from the actual Transaction Record. In most cases, this occurs if the Transaction detects an error and resends the current screen, together with an error message.

2. LU6.2 (IMS) identification

   For LU 6.2, identification is done via a message output descriptor (MOD) identifier provided by a user exit. Problems can be:

   - *No* successor or *more than one* successor defines a MOD named like the MOD identifier received. Check the MOD identifiers of the successors. (Advanced Page 3)
   - Wrong code page is specified for the connection spec (For Version 3.0b and up).

     Note: For LU6.2 connections under VisualAge, the default code page is your local code page, so you must always change it to be the host code page.

3. CICS and IMS LU2 identification

   - No literal fields selected for physical identification. If you have more than one successor, you must select fields in each successor for unique identification.
   - The physical identification of two or more successors is ambiguous. This may occur if two successors define the same fields (equal position and literal) for identification or if the identification fields selected for one successor also match fields in another one.

- The literal contains leading blanks which are stripped off. Until Version 3.0a, change the field offset so that the field begins with the first real character. Then remove the blanks from the literal. For Version 3.0b, simply turn off the Trim flag on Advanced page 3. Note that trailing blanks are no problem.

4. CICS LU2 and EPI identification

   - For CICS, literals can be changed dynamically. It is common style for CICS programmers to change field literals and attribute bytes at run time. This is why you must usually correct the parsed information to match this. Take special care when handling literals with leading blanks.

Exception: TransRecord Does Not Understand

Question Why do I get the *Bpl...TransRecord does not understand ...* exception?

Answer This occurs if you send a misspelled message to a Transaction Object. These are composite parts, which send all messages not understood to their primary part. In most cases, the primary part will be the first part you add to the transaction object—that is, a transaction record.

HLLAPI Exceptions: Harclock Exceptions

Question Why do I get the Harclock exception?

Answer This occurs when the CICS and IMS Connection writes to a screen that is in an incorrect state. The reasons for this can be:

- The same short session ID is used for several transaction objects, and more than one transaction object wants to update the screen. This leads to either both trying to update at the same time or one finding an incorrect 3270 screen for update. In this case, use Session classes to manage access to the HLLAPI resources.
- A previously run TOM left an incorrect panel state. This can occur if an error forced the previous TOM to abort, so that it was not able to return to a defined state. Your exception handlers should guarantee that each TOM reaches one of a defined set of states (either the base state or the ones the TOM should reach).

Time-out Exceptions

Question I don′t understand—why do I get a timeout using HLLAPI all the time? I can′t find any error in my setup.

Answer If HLLAPI time-outs occur with no obvious reason, ensure that you have defined a sufficient settle time in the 3270 connection spec. For example, try 1000 ms in case of Windows. The 3270 communication implementation, which uses its own notification mechanism, has been updated for Version 3.0b.

Drag-and-Drop for Windows 95

Question Using Windows 95, I have noticed a pop-up window appearing when I drop items from the connections page. Should I take any action on that window?

Answer Simply click again to finish the drop.

Screen Scraper Functionality

Question Is there some kind of screen scraper functionality in CICS and IMS Connection?

Answer Although CICS and IMS Connection is no screen scraper, we included a feature that enables customers to have some of this functionality for special purposes. Every time a Transaction Record is processed, it signals an event named #processed, together with the current Transaction Result as a parameter.

IBM Smalltalk 64 KB Method Size Limit

Question I get an error message saying that the *abtBuildInternals* method is too big. What can I do to avoid that?

Answer Using IBM Smalltalk, the maximum method size is limited to 64 KB. This may be critical for transaction objects, which store their definition to a method named #abtBuildInternals. If this happens, use external transaction records to distribute the transaction flow over more than one transaction object.

Using IMS, when a whole transaction is modeled in one format definition, even one transaction record can become too big. If this happens, spread its panels into several parts and parse them into several transaction objects.

BplBusinessObj class> > allInstances Behavior

Question What do I use BplBusinessObj class>>AllInstances for?

Answer This method is often used to query all instances of a class. Note that the implementation for business objects does not force a global garbage collect as a side effect. If you need a global garbage collection, invoke it through *System globalGarbageCollect*.

Business Object Key Alteration

Question Can I change the key of a business object? How?

Answer In general, you should avoid altering a business object's key. If you have no choice and are experienced, you can alter it by performing the following steps:

- Make sure that there is no other instance defined with the target key using:

```
Smalltalk
<class>allInstances detect: [ :aBo | aBo includes: keyDictionary ]
```

- Change the keys using setter methods.
- Rehash the set of instances using:

```
Smalltalk
<BO class> allInstances rehash.
```

- Process the piece of code using:

Smalltalk

```
[ <your code> ] critical.
```

to avoid concurrent update.

Run-Time Image Build Problems

Question I can't create a run-time image. I use phantom instance variables, could this be the problem?

Answer When using phantom instance variables to model 1 to n field-to-attribute relations, you may have problems when creating a run-time image. If setter methods are missing in the run-time image, the simplest approach is to define public interfaces for them.

Host Transaction Interface Changes

Question Do I have to change anything when the interface to the host transactions changes?

Answer Every time the host transactions interface is modified, you have to model these changes into your transaction records. There are two ways to do this:

- Model the changes using the Transaction Record Settings notebook. You will have to extract the changes necessary and do the update via the notebook pages. The advantage here is that you do not lose the other information the transaction record contains.
- Parse the macros again. You can have a set of locally modified macro files (to split fields, and the like), to which you apply the changes. After parsing, you will have to model the connection and identification information again, but the changes are consistent. The advantage here is that you only do the changes once. (But you then have to parse several times, one for every record.)

Communication Considerations

Question What are the pros and cons of the different communication setups?

Answer Here is a comparison of the various possible communication setups.

Static Communication Session Acquisition Setup

Here the same session is always used, and only one transaction can be handled at any one time. The following should not cause problems for your transaction system:

- Prototyping Setup via HLLAPI

  Use 3270 communication, specify short session IDs. Because it does not protect the communication resources, it is suitable only for single-threaded prototyping and testing. Each transaction must return to a fixed state after processing.

- Thread Serialization Setup via HLLAPI

  Use 3270 communication, specify session class. The session classes protect your communication resources, so you can have parallel use of communication resources. However, the session classes will serialize the requests, so there can be only one active thread using a certain communication resource at any one time (all others will sleep). Each transaction must return to a fix state after processing.

- Stacked Processing Setup via HLLAPI (IMS Only)

  Use 3270 communication, specify session class, make use of the #hold and #release messages. This is possible only if you are limited to conversational transactions. Think of a transaction being run in a 3270 session: The hold message will cause MFS to present a new blank screen on that terminal, so that another transaction can be started. You have to issue the release message before continuing work with your original transaction, and there can be multiple stack layers. Each transaction must return to a fixed state after processing.

Dynamic Communication Session Acquisition Setups

These are communication setups that allow parallel transaction processing for one user ID. However, this will produce the desired results only if the selected transactions are designed to run in parallel.

- CPI-C (IMS Only)

  Using CPI-C will give you dynamic session acquisition and host performance advantages. Dynamic session acquisition means that every time a session is needed, one is started automatically. This

means that you can have multiple parallel sessions. Read the limitations section in the CICS and IMS Connection documentation to decide if there is any show-stopping limitation for your setup. Note that CPIC causes very much debugging effort, so it seems suitable for production only, not for development.

- EPI (CICS Only)

  EPI also offers dynamic session acquisition. It does not have as many limitations, and causes only slightly more debugging effort compared to HLLAPI.

Chapter 5. Database

In this chapter we cover database-related questions.

Error Message When Upgrading

Question After filing in code from VisualAge for Smalltalk, Version 2 Standard to Version 3 Team, I get a debugger with ″AbtIbmDate384Field does not understand format:″ when generating run-time code for an access set.

Answer Implement a *format:* instance method for AbtIbmDate383Field that does nothing.

Ineffective Bind Command Syntax

Question The syntax of the bind command of the *VisualAge for Smalltalk for Smalltalk User's Guide* does not work for me. What do I do?

Answer Use the commands below to bind:

```
dbm connect to <database name>
dbm bind <.BND file> to database using DATETIME ISO BLOCKING ALL
```

User ID from User Profile Manager

Question How do I get the User ID from User Profile Manager?

Answer If you are using DB2 for OS/2 (and in possibly other configurations as well), you can do the following in Version 3 of VisualAge for Smalltalk:

```Smalltalk
| pf result userid type |

pf := PlatformFunction callingConvention: 'pascal16'
  function: 'UPMELOCU'
  library: 'UPM'
  parameterTypes: #(struct struct )
  returnType: #uint16.

userid := String new: 10.
type := ByteArray new: 2.

result := pf coroutineCallWith: userid asPSZ with: type.
userid inspect.
^ userid
```

For Version 2, change 'pascal16' to 'pascal'.

Password-Required Warning when Using TopLink

Tip

If you get a cryptic warning message from Q&E saying that a password is required when using VisualAge for Smalltalk with TopLink, the problem is probably related to the ODBC drivers shipped with VisualAge for Smalltalk and TopLink. TopLink ships with front-end ODBC drivers that will work with almost all back-end ODBC drivers except those shipped with VisualAge for Smalltalk. Our licensing agreement with Intersolv is for the use of the Q&E drivers that we ship as both the front-end and the back-end drivers. This is enforced through a password mechanism. Since TopLink ships their own front-end drivers, trying to use the back-end drivers shipped with VisualAge for Smalltalk causes a problem. To solve this, TopLink can use the front-end drivers provided by VisualAge for Smalltalk (not available yet) or the customer can use other back-end drivers (such as from Microsoft or Intersolv) that have a per seat run-time fee.

DB2/6000 Connection Failure on AIX

Question I get a walkback (primitive failed in: PlatformFunction...) when I try to connect to DB2/6000 from VisualAge for Smalltalk on AIX. I recently installed VisualAge for Smalltalk, Version 3a and also recently changed versions of DB2. I then backed out a later version of DB2 and reverted back to 2.1. The connection to the database manager fails; the database DLL fails to load (is not found). What's wrong?

Answer The libpath statement in the ABT file should be updated to include "/usr/lpp/db2_02_01/lib" after the "/usr/visualage/bin" entry. This is not consistent on all configurations and may have something to do with the version of DB2/6000 you are using.

Note

There is a known problem with DB2/6000 Version 2.1 that may explain this situation. The external symptom of the DB2/6000 problem is the inability to load DLLs, archived libraries, or both. The problem is caused by an old or back-level version of C run-time support shipped with the DB2/6000 V2 image. The following circumvention was extracted from a DB2/6000 PMR. Find a newer version of libC.a. Check in usr/lpp/xlC/lib. The size of libC.a that works is 410 KB. When you find a correct version of libC.a, copy or link it into the /usr/lpp/db2_02_01/lib directory. Make a backup copy of each prior to the link. Run: $ln -fs /usr/lpp/xlc/lib/lib C.a/usr/lpp/db2_02_01/lib/libC.a

Handling Errors in Database Code

Tip There are several ways to do error checking for SQL in VisualAge. One is to connect a block to the *errorBlock* attribute of the database query using an attribute to script connection. The script can look something like:

Smalltalk

```
handleError

 ^[ :error | ErrorView newPart valueofAttributeNamed: #String
  put: (error printString);
   openWidget. ]
```

If you want to check for a specific error code you can do something like:

```
Smalltalk
handleError2

 ^[ :error |

 |code|

 code := AbtDbmSystem activeDatabaseMgr sqlca at: 'sqlcode'.
 self partAttributeValue: #(#Text6 #object) put: code.

 (code = 100)
  ifTrue: [ErrorView newPart valueOfAttributeNamed: #String
        put: 'SQL 100 Error';
     openWidget]
 ].
```

SQL Error 30081n in DB/2 2.1

Question I received SQL Error 30081n when I was in DB/2, Version 2.1. What do I do?

Answer I had a customer who solved this problem by going to Communications Manager and reconnecting to the database server. Apparently the link between the database server and database client was inactive. Reactivating the connection made the error disappear.

Call-Level Interface

Question The feature list includes an entry for IBM CLI Database. What's that?

Answer That's a good question, and I'm glad you asked. Version 3.0a was mainly a port to Windows 95 and NT. We didn't plan on putting in new features, or updating our documentation either. But one feature did get in, and that's the IBM CLI Database. Unfortunately, this feature has minimal documentation (its all in a readme file). So, I will tell you about it.

The IBM CLI Database feature is a reimplementation of the IBM Database Manager, bringing our database manager count to four:

1. IBM Database 2 (the original)
2. IBM Database 2 - CLI
3. ODBC

4. Oracle (Native)

Our original DB2 support used a DLL that we provided. That DLL used embedded SQL. It didn't support multiple connections, and it didn't handle concurrent applications well. It needed to be ported to Windows 95 and Windows NT. Furthermore, it is not possible to support multiple concurrent connections using embedded SQL (you can have multiple connections, but only one active connection). Faced with these problems, we decided it was time to change interfaces.

DB2 provides a second dynamic interface, the Call Level Interface (CLI), which supports multiple concurrent connections. The CLI DLL is shipped with each DB2/2 client, so our customers already have it. It's designed for concurrency of applications. It supports distributed units of work (multiple connections, one transaction). It's available on all platforms. So,... we now support the CLI.

The CLI should have the same public interface as the original DB2 support, so migration is easy. It is necessary to use AbtEditDatabaseMigrationApp to migrate any of the visual database query parts, however. (This was documented in VisualAge for Smalltalk, Version 3.0, you load the application and execute *AbtDatabaseMigrationView new openWidget*).

The CLI feature is available on all platforms (the original DB2 support has not been ported to Windows 95 or NT), and it is the implementation that we will be using in the future. The original DB2 support will eventually be dropped.

Binding Problem with DB2/6000 on AIX

Question What, exactly, do I need to write a stand-alone VisualAge for Smalltalk application on AIX that works with DB2/6000? I have installed DB2/6000 Version 1.2 on AIX 4.1.3, but I cannot successfully bind. I get a message ″SQL0033N ″/usr/visualage/abt/abtdx30.bnd″ is not a valid bind file″. I get the same error when I try using abtdsx30.bnd as well.

Answer The default *.bnd and *.w file installed for VisualAge for Smalltalk, Version 3.0a on AIX, is compatible with DB2/6000 2.1. To use DB2/6000 1.2, copy the files abtdx30.bnd and abtdx30.w to your local VisualAge directory from /usr/cdrom/vast/install/fileins. This will allow you to run DB2/6000 1.2

If all VisualAge for Smalltalk users use DB2/6000 1.2, then copy abtdx30.bnd to /usr/visualage/abt and copy abtdx30.w to /usr/visualage/bin. Both sets of files have the same names, so please save a copy of the 2.1 files. Also

notice that there is a file-in for DB2/6000 1.2 (db2v120.st). You can use the database samples to perform the bind.

OS Error 126 Connecting to Oracle

Tip

Oracle changes the DLL names with every release. The DLL name is hard-coded in our methods and therefore the customer may get a ″can′t find DLL″ condition when connecting to Oracle. To fix, change the DLL name in private class method *defaultLibraryName* in AbtOracleDatabaseManager and then execute ″AbtOracleDatabaseManager buildPlatformFunctionsDictionary″ to make the change in the class variable. Here are complete instructions to make appropriate changes to VisualAge for Smalltalk code:

1. From Smalltalk Tools/System/Change User, make yourself Library Supervisor.
2. From Smalltalk **Tools/Manage Applications**, find application **AbtDbmOracleBaseApp** and select it.
3. From **Applications**, select **Create New Edition**.
4. Find application **AbtDbmOracleWinPlatformInterfaceSubapp** below **AbtDbmOracleBaseApp** and select it.
5. From **Applications**, **Create New Edition**.
6. In the next pane, choose class **AbtOracleDatabaseManager**, and double-click on it to open a browser.
7. In this new browser, choose everything in the second pane, and click on the instance button below the third pane so that it says *class* and click on the public button below the fourth pane so that says *private*. Select everything in the thrid pane and you will see the method defaultLibraryName in the fourth pane.
8. Click on *defaultLibraryName* and change the code in the lower pane to indicate your DLL for Oracle.

   For example, to reflect your DLL name, change ″orant71.dll″ in the code below:

   Smalltalk

   ```
   IsWinNT
     ifTrue: [^'orant71.dll']
   ```

9. Save the changes to the method with the pop-up menu, and close the browser.

10. From the Application Manager, version and release your class to a name you will recognize (Classes/Version/Release All).
11. Click on **AbtDbmOracleWinPlatformInterfaceSubapp** and version and release it to the same name (Applications/Version and then Applications/Subapplications/Release).
12. Click on **AbtDbmOracleBaseApp** and version it (Applications/ Version).
13. Change back to your user ID.
14. Other coworkers wanting to take advantage of the fix can now load the new edition of the application into their image.
15. After doing this, type *AbtOracleDatabaseManager buildPlatformFunctionsDictionary*. in the Transcript, select it and execute.

SQLSTATE 37000 Error with ODBC Driver

Question When using the VisualAge for Smalltalk ODBC driver with MicrosoftAccess, I get ″Error string {SQLSTATE 37000 - {Microsoft} {ODBC Access Driver} 10014 in query expression {Native Error 3100}} Why?

Answer The cause of this problem is that Microsoft Access does not support the use of the ″for update″ clause. Because of this restriction, you must uncheck the checkbox ″Lock row on edit″ on the Update notebook page when looking at the settings for your database query.

Microsoft Access Drivers

Tip To get updated drivers for Microsoft Access:

1. Point your favorite Web browswer at www.microsoft.com.
2. Select the Support link.
3. Then select the **Free Software from Microsoft Software Library** link.
4. Under the **Explore:** option, select **Access**.
5. For the Search field, type *ODBC drivers*.
6. Hit the **Go** button.
7. Download the file from the ″ODBC Drivers for Win95 Applications″ link.

SQLSTATE S1010 Error with ODBC DB2/2 Driver

Question I got the following error message: [SQLSTATE=S1010 - [INTERSOLV][ODBC DB2/2 driver]Function sequence error. [Native Error=0]] What do I do?

Answer Try the following:

Go into the VisualAge for Smalltalk DB2/2 driver's setup from the ODBC Administrator. At the bottom, there is an option for Cursor Behavior. The default value is 0 - Close. Try choosing 1 - Preserve.

The DB2 Driver, by default, does not use "with hold cursors." In other words, if a commit is done, the cursors are no longer valid. If you set the cursor behavior option to 1 - Preserve, it causes the driver to be used "with hold cursors," which causes the cursor to remain valid after a commit has occured.

This is a DB2/2 behavior. In DB2/2, when a commit is done, the cursors become invalid. In our IBM database code, when a commit is done, we just flush the cache to avoid the problem with invalid cursors. But in ODBC, it does not make sense to flush the cache for a commit since this behavior is unique to DB2/2. So, this setting needs to be set in the driver to cause it to use "with hold cursors" so that the cursors do not become invalid after a commit.

Primitive Failed—OS Error 1

Question I am trying to connect VisualAge for Smalltalk to DB2/6000. I run AIX and DB2 2.1. I get a walkback saying: Error string: Primitive failed in: PlatformFunction>>#callWith:with:with: due to OS error1. I have the impression that DB2 connection outside VisualAge is working.

However, I would like to check my path setup to verify that everything is correct. What do I have to do to check this?

Answer To ensure that your environment is correct, enter "db2" at a command line. If their environment is correct, this command will enter you into the db2 command-line processor. If this command fails, enter "set" at a command line. After entering "set" ensure that there is an entry in your path for db2 (/home/db2/sqllib/bin). If the "db2" command succeeds, then take at look at

/usr/visualage/bin and ensure that abtdx30.w is present. Finally, try to execute the DB2LN command.

Migrating from DB2 V1.2 to DB2 V2

Tip If you have developed an application in VisualAge for Smalltalk that accesses DB2 tables version 1.2, and you want to migrate from DB2 1.2 to DB2 2, it should have no impact on your application. The only thing I can think of that you might want to change is if you have any references to the class AbtIbmDb22DatabaseConnection, you may need to change them to AbtIbmDb22CSDatabaseConnection, although I think the code would work even if you don't change.

Building a Dynamic Where Clause

Question I am trying to build a dynamic where clause. The data for the clause comes from two text entry fields. One field states the customer's name and the other states the customer's identification number. My script looks as follows:

Smalltalk

```
initializeWhereClause
"Private - initialize the where clause of the SQL query
looking for customer name and ID"

| where |
where := 'WHERE CUSTOMER.CUST_NAME = ',
  (self subpartNamed: 'CustomerName') value,
  ' AND CUSTOMER.CUSTID = ',
  (self subpartNamed: 'CustomerID') value.
```

What I want is:

SQL

```
WHERE customer.cust_name = 'John Doe'
  AND customer.custid = 102
```

What I get is:

```SQL
WHERE customer.cust_name = John Doe
  AND customer.custid = 102
```

How can I create a robust method that will handle name punctuation, for example O′Brian, in the correct way as well?

Answer

The problem you are having is you are including an extra quote before and after the name string. Try the following code snippet using our sample tables:

```Smalltalk
| activeConnection querySpec result resultCollection temp aName |
resultCollection := OrderedCollection new.
aName := 'O''Brien       '.
activeConnection := (AbtDbmSystem activeDatabaseConnection).
querySpec := (AbtQuerySpec new) statement:
  'select * from STAFF where NAME = ', aName printString.
result := activeConnection resultTableFromQuerySpec: querySpec.
result do: [:row | resultCollection add: (row asString)].
^resultCollection.
```

Another way to write the query would be this code:

```Smalltalk
| activeConnection querySpec result resultCollection temp aName par |
resultCollection := OrderedCollection new.
aName := 'O''Brien       '.
querySpec := (AbtQuerySpec new) statement:
  'select * from STAFF where NAME = :name'.
par := Dictionary new.
par at: 'name' put: aName.
activeConnection := (AbtDbmSystem activeDatabaseConnection).
result := activeConnection
  resultTableFromQuerySpec: querySpec
  withValues: par.
result do: [:row | resultCollection add: (row asString)].
^resultCollection.
```

In this way you can pass the values of all the host variables using a dictionary. Something that can help improve performance is the use of the Access Set (provided that you are using VisualAge for Smalltalk Version 3). In other words, you can define your select query using the interactive tool and store it in one Access Set (let's call it MyAccessSet) as "MyQuery." Once you have done this, you can retrieve the query with the following code:

```
Smalltalk

query := MyAccessSet runtimeQuerySpecNamed: #MyQuery
```

This version of the code usually runs faster, since the query stored in the Access Set contains a definition of the host variables (type, length and so on). You can get this definition by sending the method hostVarsShape to the query specification (instance of class AbtQuerySpec). Therefore the final code becomes:

```
Smalltalk

| activeConnection query result resultCollection temp aName par |
 resultCollection := OrderedCollection new.
 aName := 'O''Brien       '.
 query := MyAccessSet runtimeQuerySpecNamed: #MyQuery.
 par := Dictionary new.
 par at: 'name' put: aName.
 activeConnection := (AbtDbmSystem activeDatabaseConnection).
 result := activeConnection
   resultTableFromQuerySpec: query
   withValues: par.
 result do: [:row | resultCollection add: (row asString)].
 ^resultCollection.
```

ODBC using Microsoft-JET Drivers

Question I'm trying to use VisualAge for Smalltalk ODBC parts to access a FoxPro file through the Microsoft JET/Dbase/FoxPro driver. Unfortunately, it seems Microsoft does not support 'FOR UPDATE OF...' in a SELECT statement. Is there any easy workaround in VisualAge, or do I have to get a different ODBC driver?

Answer You are correct that the Microsoft Jet ODBC Driver does not support the SELECT .. FOR UPDATE OF... statement, which is used to lock the row selected. There are two options you can use:

1. You can go to the Update page on the MultiRowQuery settings page and turn (Lock row on edit) off. Of course, the locking scheme will be in your control.
2. You can use the dBase driver that is shipped with VisualAge for Smalltalk and go to the driver settings in the ODBC Administrator and choose FoxPro1 or FoxPro25 for the CreateType and fox for the Lock Compatibility. More information can be found in the online document: *ODBC Drivers Reference*.

Database Log-on Prompt

Question The log-on prompt is really beginning to annoy me. It seems to pop up for every database operation, even though I have prompt turned off. The only thing I can think of that might be causing this is that I have multiple database access sets defined. When I added the connection to the application, it only added to the one access set. I manually added it to the other, but I believe it is prompting me every time it has to switch between access sets. I want the thing turned off completely since I am writing an application that will run unattended.

Answer If you are using ODBC, then be aware that it has a prompting mechanism of its own, so if you do not supply sufficient log-on information, the ODBC prompt will pop up on a connection attempt.

VisualAge for Smalltalk will attempt to establish a new database connection only if there are no existing connections with the specified connection alias. Are you certain that the connection alias for all of your parts is the same? Does your application disconnect the connection? Be aware that the #close action of the database query parts will cause a disconnect. When you test your application, use the Database connections view from the VisualAge for Smalltalk organizer to monitor the status of your database connections (the connection view refreshes each time it receives focus).

It is not necessary to add the connection specification to multiple access sets within the same application. The database query parts can use any connection specification that is visible to the application. This includes connection specifications that are defined in access sets that are owned by or are prerequisistes for the application.

Since your application is to run unattended, you might consider using the method below to save your log-on specification. When you run this code, your log-on specification will be active for as long as the image remains active.

Smalltalk

```
" For ODBC, use the data source name as the server "
AbtDatabaseLogonSpec
  id: 'myid'
  password: 'mypassword'
  server: 'servername' alias: 'Alias1'


" If you would like to 'unregister' the logon spec, execute the following: "
AbtDbmSystem
  removeLogonSpecWithAlias: 'Alias1'
```

Database Log-on Prompt after Migrating to Version 3

Question I migrated an application from VisualAge for Smalltalk, Version 2.0 to Version 3.0 and after setting up a database connection with prompt set to "no," I get a log-on prompt from VisualAge asking for user ID, password, and node every time I connect to the database (DB2). I don't want this prompt.

I'm not sure if this is anything to do with the fact that the application uses "connect" which, the user's guide informed me, is sort of just left in for compatibility. How do I lose the prompt?

Answer If you use *openDatabaseNamed:* in your application, try removing it. That should eliminate the prompt you are getting. Try something like the following. (Note that there's also a change in a variable name for a connection that's really a database manager.)

```
Smalltalk

"Selects rows from a table"
| querySpec result resultCollection connection manager |
resultCollection := OrderedCollection new.
manager := AbtDbmSystem
  activeDatabaseMgrWithName: #AbtIbmDatabaseManager.
connection := manager activeConnection.
querySpec := (AbtQuerySpec new)
  statement: 'SELECT * FROM STAFF'.
result := connection resultTableFromQuerySpec: querySpec.
result do: [:eachRow |
  resultCollection add: (eachRow asString)].
^resultCollection.
```

Native Oracle and Data Types

Question I would like to use VisualAge for Smalltalk Professional on Windows 95 and NT platforms and hence I would like to know if there are any restrictions on using native Oracle support on Oracle data types accessible, specifically Long and LongRow?

Answer We are not aware of any restrictions or limitations when using these data types. One piece of information you might find useful is that default buffer size for these datatypes is stored in a class variable and can be changed. The default is 32767. To change the default, execute:

```
Smalltalk

AbtOracleLongField bufferSize: <newSize>
```

The database resource cache for any active database connections should be cleared out before proceeding. You can do this as follows:

```
Smalltalk

(AbtDbmSystem
  activeDatabaseConnectionWithAlias: 'Oracle1' ) flushCache
```

There is an undocumented interface that can be used to fetch portions of LONG/LONG ROW fields. Fields of this type are often huge, and it is not expedient to fetch the entire field with one trip across the network. Below you find code to demonstrate how a partial fetch can be accomplished:

Smalltalk

```
| tbl row rt longBuffer |
tbl := AbtDbmSystem
  activeDatabaseConnection openTableNamed: 'TLONG2'.
row := tbl emptyRow.
rt :=  AbtDbmSystem
  activeDatabaseConnection resultTableFromQuerySpec:
    ('select * from TLONG2 where COL2 = ', 'abc' printString )
       abrAsQuerySpec.
rt next.

" Fetch only a portion of the long field into longBuffer "
longBuffer := rt cursor
  currentLongFieldNamed: 'COL1'
  from: 31990
  to: 33200
  ifError: [].
rt close.

longBuffer inspect
```

Local Log-on

Question I would like to do an automatic log-on to the database server, because I don't want the User Profile Manager (UPM) to ask for my user ID. Does anyone have a code snippet describing how to do that?

Answer The following fileout of a class has some UPM methods in it that will allow you to do a local log-on and log-off in VisualAge for Smalltalk, Version 3.0. Here is how we use it in our application (CSCCUPM is a subclass of CSCCObject):

Smalltalk

```
upm := CSCCUPM new.
rc := upm
  upmLogon: userID
  password: password.
rc isAbtError ifTrue: [
  "A UPM logon error occurred return the error to the caller"
  ^]

logonSpec := AbtDatabaseLogonSpec
  id: userID
  password: password
  server: nil.
conSpec := AbtDatabaseConnectionSpec
  forDbmClass: #AbtIbmDatabaseManager
  databaseName: database.
conSpec promptEnabled: false.

rc := conSpec
  connectUsingAlias: 'CSC ConSpec'
  logonSpec: logonSpec.
```

Text Database

Tip

This is an example of how to create a text database using the ODBC Text driver. Before running this script, you need to have already defined a data source for the text driver using the ODBC Administrator (using the ODBC Sample Launcher). Use this:

Smalltalk

```
 | textDatabase  result  def tableName |

"This is the name of the table"
 tableName := 'TEST'.

"Add the appropriate DSN for databaseName:"
 textDatabase := (AbtDatabaseConnectionSpec
  forDbmClass: AbtOdbcDatabaseManager
   databaseName: 'Test' ) connectUsingAlias: 'test'
   logonSpec: (AbtDatabaseLogonSpec new
    id: '';
    password: '';
    server: '').

def := '(NAME    VARCHAR(30) ,',
 ' STREET    VARCHAR(30),',
 ' CITY     VARCHAR(20),',
 ' STATE    VARCHAR(2),',
 ' ZIP_CODE   VARCHAR(5))'.

result := textDatabase
 createTableNamed: tableName
 definition: def
 ifError: [
  AbtDbmSystem activeDatabaseMgr errorBlock. ]
 textDatabase disconnect.
```

Reusing a Cursor

Question Is there any method in the Multirow Query part that I can use to prepare a query only once and then execute the query several times with different host- variables? I understand that I can't use a dynamic "Where" clause. My reason for asking is that, at least in the AS/400 field, I pay a severe penalty every time I run the query (for example, for every row when filling a container details part).

Answer You don't have to do this, it's already implemented as described. When a cursor is prepared, it gets saved in a cache (resourceCache instance variable on a database connection object). Each time you use #executeQuery, the cache is searched to see if there is a prepared cursor available to be reused. If a cursor is open, it can't be reused; you can't open the same cursor twice. Once the cache size reaches 10, the least recently used cursor is thrown out before adding a new one.

OS/2 ODBC Problems

Question I have come accross quite a few problems when trying to use VisualAge for Smalltalk, Version 3.0 ODBC support under OS/2. First of all, an attempt to create VisualAge Text data source causes a trap D every now and then (not regularly).

The second problem is that an attempt to add a VisualAge DB2/2 based data source results in an error message: "INTERSOLV • ODBCADM • Not able to add a datasource."

The last problem I have is with the ODBCADM executable: it freezes the window if the program is started from the OS/2 command prompt. The window seems to be alive—the cursor blinks—but I can't type anything. The ODBC feature is installed, samples as well, drivers installed, all without any error messages.

Answer

The first problem may occur if you had a previous version of ODBC installed on your system, or had the previous version of VisualAge for Smalltalk with ODBC support installed. The best thing to do is to delete the files named: IB*04.DLL, ODBC.DLL, ODBCINST.DLL, and reinstall VisualAge for Smalltalk again to make sure you are using the latest level of drivers.

The second problem will occur when the DB2/2 Driver (IBDB207.DLL) can't find the required DB/2 Client Dlls in the LIBPATH. If you have a tool called chkdll32 you can check to see what DLLs that driver requires to load.

For the last problem, try the following: the ODBCADM Executable that was provided to us from a third party vendor trashes your OS/2 Window. Try typing: Start ODBCADM at the OS/2 command prompt.

Connecting to DB2/400 with ODBC

Question I would like to connect to DB2/400 using the ODBC driver of Client Access/400 Optimized for OS/2. I couldn't find any documentation on this and I would appreciate some help.

Answer If you want to use the ODBC driver with VisualAge for Smalltalk, you will have to install two PTF's for Client Access/400 (product 5763XG1 V3R1M1):

- SF28583
- SF28840

This will update your Client Access/400 Optimized for OS/2. You will have the correct DLLs (EHNODBC2.DLL and EHNSTP2.DLL in the CAOS2 directory). If you installed ODBC support on your VisualAge for Smalltalk environment, you should have an ODBC administrator program in the DLL subdirectory of your main VisualAge directory, called ODBCADM.EXE. This program will enable you to configure datasources that use the ODBC driver of Client Access. Specify which machine you want to connect to, and which libraries you want to use. Leave the other parameters at their default value.

You have now created an ODBC data source that you can use like any other data source in VisualAge for Smalltalk. If you want more information on using data sources and database parts, check out Chapter 2 of the *VisualAge for Smalltalk User's Guide*, Version 3. This chapter will tell you more about database support. The database in the example provided is DB2/2, but basically it is the same as an ODBC data source. They both can execute SQL statements.

Retrieving Current Date from DB2/2

Question When I execute a SELECT to the database to get the current date, VisualAge for Smalltalk is changing the date format from a 4-digit year to a 2-digit year. Is there any way to override this conversion?

I have been told that the correct date format is returned from DB2. VisualAge for Smalltalk converts the date to a Date object. When the date is then accessed via printString, a 2-digit year is presented as the default format. For my situation this is not an acceptable response. The date should be presented in the same form that it is returned by DB2. Without modifying Date or the converter class that Date uses, what is the correct method for obtaining the current date from DB2?

Answer The problem comes down to a pool dictionary NlsGlobals, which, in turn, contains an entry for CurrentLCTime, an instance of LCTime. LCTime has a method called #dFmt: that takes a string as parameter, as in dFmt: '%d/%m/%Y'. So using:

```
Smalltalk
(NlsGlobals at: 'CurrentLCTime') dFmt: '%d/%m/%Y'
```

changes to 4-digit years in printing, sending '%d/%m/%y' switches to 2-digit year printing.

To answer the second question, you won't receive back a Date object from the database. Instead you get back a AbtRecord of type 1 that is just a dictionary that contains the date at: '1'. By sending it the at: message you can get the Date object.

A date object has an instance variable called *year* that contains a 4-digit year. When you inspect or print a Date object, the year is convert from 4 digits to 2. For example, inspect the following: Date today. As you can see, the year is truncated from 1997 to 97. The 4-digit year can be accessed by sending the year message to the date object. When you execute a SELECT statement and retrieve the rows from the result table, you get an ordered collection of AbtIbmRow objects. Each row object is itself made up of objects. If your SELECT statement includes a column defined as a DATE data type, then the row objects will contain date objects. To access a date object, send the at: message to the row object. The argument passed is the name of the column that contains the date value. For example, suppose you executed the following SQL statement:

```
SQL
SELECT CUSTOMER.NAME, CURRENT DATE DATE FROM CUSTOMER.
```

To get a date you would execute the following statement:

```
Smalltalk
rowObject at: 'DATE'.
```

Once you have the date object, you can access the 4-digit year.

Database Portability

Question Suppose I develop an application with VisualAge for Smalltalk that contains lots of queries to DB2. What effort would it require to convert it to another database (for example Oracle), if I use:

- Native SQL?
- ODBC?

Answer A tool is provided in VisualAge for Smalltalk, Version 3, which helps users migrate their Version 2 database parts to Version 3. This tool also helps users to migrate their query parts to use different connection specifications. See page 429 of the *VisualAge for Smalltalk User's Guide* for additional information about this tool.

The general steps for migrating your query parts would be as follows:

1. Modify an existing connection specification for your application (or create a new specification) so that it contains information about the desired (target) database connection;
2. Follow instructions on page 429 of the *User's Guide* to execute the migration tool for the parts that require migration.

Migration will be possible if the table names and column names for your target database connection match those for your original connection.

Canceling a Database Call

Question I would like to know if there is an ability for the end-user application to cancel a database call. The scenario is that a user sets up a long-running process and then decides to abort the transaction request.

I can specify threaded database calls from the preferences notebook. One approach might be to end the thread as a means of canceling the database transaction. I assume the thread could be terminated by an external call interface and using a facility of the platform's operating system. What support is available in the database classes to cancel a database request?

Answer Native Oracle support contains an API called ″break″ that will asynchronously terminate any long-running operations for a database connection. This method can be executed as follows:

Smalltalk

```
(AbtDbmSystem activeDatabaseConnectionWithAlias: 'Oracle' ) break
```

ODBC and IBM database support do not currently implement any similar functionality. In answer to the question regarding threads, threaded database calls are executed with a threadKey equivalent to the AbtDatabaseConnection instance that executes the DB call. Therefore, a thread could be terminated by executing the following code:

Smalltalk

```
AbtThreadManager terminateThread:
(AbtDbmSystem activeDatabaseConnectionWithAlias: 'x').
```

Establishing a Database Connection via Smalltalk Code

Tip Below you find an example of the recommended VisualAge for Smalltalk, Version 3, approach for establishing a database connection via Smalltalk code. (The recommended steps for establishing a database connection have changed. The old—Version 2—technique for connecting is still supported, but is slightly less flexible.) The code is as follows:

Smalltalk

```
| dbSpec1 dbConnection1 |

dbSpec1 := AbtDatabaseConnectionSpec
  forDbmClass: AbtOdbcDatabaseManager
  dataSourceName: 'Local'.

"activate the connection described by dbSpec1 passing in required
logon information "
dbConnection1 := dbSpec1
  connectUsingAlias: 'Local'
  logonSpec: ( AbtDatabaseLogonSpec new
    id: 'USERID' ;
    password: 'PASSWORD' ;
    server: 'Local' ;
    yourself ).
```

The approach above is useful if you are using Smalltalk code to connect. However, the VisualAge for Smalltalk visual database parts are designed to prompt for log-on information each time a new connection is established. There is a simple way to override this behavior. A programmer can "register" log-on information, which remains active for as long as the image is active (the information is cleared when the image is restarted). For example,

Smalltalk

```
AbtDbmSystem registerLogonSpec:
  logonSpec: ( AbtDatabaseLogonSpec new
    id: 'USERID' ;
    password: 'PASSWORD' ;
    server: 'Local' ;
    yourself ) withAlias: 'Local'.
```

After the above statement is executed, the supplied log-on specificationswill be used any time you attempt to activate the connection with alias Local. The #removeLogonSpecWithAlias: can be used to undo the above action.

Non-ANSI SQL Support

Tip Can VisualAge be used to generate non-ANSI SQL? If not, could I manually create a non-ANSI SQL query for a database part?

Question The VisualAge for Smalltalk SQL Editor was built based on ANSI SQL and cannot be used to build non-ANSI SQL. However, you could manually create a non-ANSI SQL query for a database part. We know of no limitations in allowable syntax for SQL statements. In some cases, it may be possible to build your initial ANSI query using the SQL Editor, then manually edit the query to add any non-ANSI functionality. VisualAge for Smalltalk merely passes your query string to the database connection. No syntax validation is done within VisualAge itself.

Binding to Database

Question I am trying to create my first query with VisualAge for Smalltalk, Version 3, using a DB2/6000 database. I have successfully created an access set and alias. When I look at the Database connection specifications window, it shows my database to be active. When I attempt to create a query in the Multirow Query Settings window, I get: SQL0805N ″NULLID.ABTDC30 is not found.″ What does this mean?

Answer This means that you did not bind to your database. You can look at the instructions in Appendix C of the *VisualAge for Smalltalk User's Guide*, or you can use the AbtSampleLauncherView in AbtDatabaseSamples to bind to your database. When you run this sample view, you can choose the Tools page of the notebook. There will be a button that says **Bind To Database**. Click this button. The database alias name that you are trying to bind to (if you have an active database connection) should show up in the list. Highlight it, and click the **Bind to Database now** button.

Using a Wild Card with Host Variables

Question I'm executing a query with the following structure and it works fine:

```
SQL
SELECT * FROM table WHERE name LIKE 'SOU%'
```

However, if I change the predicate member as follows: connect a text part to :varHost and enter SOU% as the text part value, no action is performed, that is, nothing is displayed.

```
SQL

SELECT * FROM table WHERE name LIKE :varHost
```

I would like to know why it works this way, or how I can use the wild character *'%'* with host variables?

Answer The problem should be solved if you use a VARCHAR host variable rather than a fixed character field. When constructing default host variable definitions, VisualAge for Smalltalk bases the definition on the table shape. For example:

```
SQL

SELECT * FROM table1 WHERE field1 LIKE :hostVar1
```

When using the VisualAge for Smalltalk SQL editor, the above query would build a default host variable definition for *hostVar1* based on the column definition for *field1.* You can use the host variables option on the settings view to modify the shape of your host variables.

The reason Fixed character host variable definitions do not work as you might expect for this case is that VisualAge for Smalltalk pads fixed character fields with spaces. On the other hand, VARCHAR fields are padded with nulls. So, if *hostVar1* is a FixedCharacter field with size 10, substituting the value *'abc%'* for *hostVar1* would result in the following query:

```
SQL

"Get all rows from table1 where field1
starts with 'abc' and ends with 6 spaces"

SELECT * FROM table1 WHERE field1 LIKE 'abc%'
```

Using a VARCHAR field with size 10 would result in the following query (as expected):

```
Smalltalk

"Get all rows from table1 where field1 starts with 'abc'"

SELECT * FROM table1 WHERE field1 LIKE 'abc%'
```

Bypass the Error Prompt in DB2/2

Question How can I prevent the error prompter from popping up when a query error occurs? We intend to use a touch-screen and want to minimize the interaction between the end user and the system.

Answer Each query part has its own #errorBlock attribute, which can be individually set prior to query execution. The query parts circumvent the default error block set in the active database manager instance by passing in an errorBlock when executing database API calls. Prior to executing a query, you can set its errorBlock in a script:

```
Smalltalk

(self subpartNamed: 'Multiple Row Query1')
  errorBlock: < your block >.
```

Another thing you can do is to create a VisualAge for Smalltalk event-to-script connection to a method which returns the errorBlock and connect the result of the connection to the #errorBlock attribute of your query part.

Delete and Create Rows

Question I tried the newRow and deleteRow action in VisualAge for Smalltalk, Version 3.0, on a resultTable torn off from a multirow query. While running the application, it turned out that the actions didn't work. Are the *newRow* and *deleteRow* messages not supported?

Answer One possibility is that the query is set to read-only. Some queries that you write can only be read-only. Look at the Update page of the query settings to see if it is set to read-only. For example, if the query includes computed columns, or is a SELECT DISTINCT, it must be read only. If VisualAge for Smalltalk determines that the query is read-only, you will see a message on the first update page of the settings view.

Handling SQL Statement

Tip Using the VisualAge for Smalltalk *SQL Statement* part, you can use the Manual create and Manual edit options to type in the text of a SQL statement. When executing statements of this type, VisualAge passes the statement to the database connection instance. As long as the connection instance can handle the statement, there should be no problem. The Smalltalk script to accomplish this nonvisually is as follows:

Smalltalk

```
| connection |
connection :=
  AbtDbmSystem activeDatabaseConnectionWithAlias: 'MyConnection'.

connection executeSQLStatement:
  'CREATE NICKNAME BOB FOR TABLE STEVE'.

"any non-SELECT SQL statement "
```

Database Operations on Separate OS/2 Thread

Question Is it possible to configure VisualAge for Smalltalk Version 3 to have long-running database queries on a separate OS/2 thread so as not to freeze the end user interface?

Answer From the VisualAge for Smalltalk Organizer, select **Options** → **preferences**. Select the **Database** page, and from there you can select **Thread all database calls**.

Hard-Coded Database Name

Question I noticed that the connection specification has the database name hard-coded in it. I thought that VisualAge for Smalltalk would let me develop my application independent of a specific database.

This independence would be helpful if I need to use one database to develop and test my application, and another for production use of my application. Otherwise, I would have to create another connection specification for run time. So, my idea is to modify the alias accessor method in order to put in an indirection for the database name in order to read it from an INI file. Would this be the right way?

Answer The database connection specification provides a level of indirection that allows you to describe your database connection separately from your database query parts. The connection alias in the query parts provides a reference to the actual specifications that are retained in an access set.

The connection specification gives the programmer the ability to quickly change the database name for all of their parts by merely modifying the connection specification. So, a programmer could develop and test using database1, but change the specification to database2 prior to packaging. This level of flexibility is sufficient for most customers.

As is pointed out, the database name is hard-coded in the connection specification; therefore, it is not possible to dynamically determine and set the database name at run time using the default connection specification scheme.

The idea of changing the accessor method for the connection specification is a good way to accomplish your objective of dynamically setting the database name at run time. The modified specification method would look something like:

Smalltalk

```
^AbtDatabaseConnectionSpec new
  dbmClass: AbtIbmDatabaseManager;
  dsn: MyInitializeClass defaultDatabaseName ; " << Changed "
  promptEnabled: true;
  yourself
```

Be warned that if you change the connection specification using the VisualAge for Smalltalk Database connections view, the specification will be replaced with a hard-coded database name. Any changes to the databaseName should be done using your initialization class (#MyInitializeClass in the example above).

Locked Rows on Database Tables

Tip

If you update a database table, the row in the database will be locked until a commit is done. The database query part has an action named *executeQueryAsTransaction,* which will do the commit as soon as the update gets done. Another possibility is the *commitUnitOfWork* action, if you want the commit to happen separately.

Sharing Queries Between Applications

Tip

It is possible to share queries between various applications by sharing the access set. The only thing you need to do is to make your application a prerequisite of the other so you have visibility of the other's access sets.

To change the prerequisites for an application, you can do the following from the VisualAge for Smalltalk Organizer:

- Select your application.
- Click **Applications** → **Make executable...** from the organizer menu
- Change the prerequisites as required
- Click **Cancel** to exit the Make executable window

Application prerequisites can also be changed using several of the Smalltalk Tools browsers from the **Transcript** menu.

Connecting from OS/2 Client to DB2/6000

Question

I am starting a project using VisualAge for Smalltalk, Version 3.0. I have put together a system architecture plan. I have a question concerning the database access. What would be the best way to connect from a VisualAge 3.0 OS/2 client to DB2/6000? I've heard about CAE/2, and DDCS/2, but I'm not sure which is better. I would appreciate some advice.

Answer Many developers/testers use CAE/2 to connect to DB2/6000 via TCP/IP. DDCS/2 is not needed to connect to DB2/6000, it is needed only if you plan on connecting to DB2/400, DB2/VM or DB2/MVS.

Outer Join Statements

Question Does VisualAge for Smalltalk support outer join statements?

Answer There are no limitations on executing outer join statements from VisualAge for Smalltalk. It merely passes the statement along to DB2/2 and iterates the results. However, queries of this type must be built manually without assistance of the VisualAge SQL editor.

Quick Form and Stored Procedures

Tip You can use the quick form option for stored procedure parts. You will notice a slightly different behavior when doing a quick form of the *self* attribute. For the query parts, doing a quick form of *self* results in a quick form of all host variables for the query. A quick form of the *self* attribute of the *Stored Procedure* part results in a quick form of all basic attributes for the part (for example, host variables, connectionAlias, databaseName, and so on). For the *Stored Procedure* part, it is probably simpler to use the quick form for each individual attribute rather than using the *self* attribute.

Using a Host Variable for IN Clause

Question I would like to build a database query which uses an IN clause. I tried to use a host variable that contains multiple values. My SQL code looks like this:

```
SQL
SELECT AA, BB, CC WHERE KEY_FIELD IN ( :hostVar )
```

What I need is a host variable so that the SQL would equate to:

```
SQL
SELECT AA, BB, CC WHERE KEY_FIELD IN ( 'aa', 'bb', 'cc' )
```

Is there another workaround? I have to allow for any number of values within the parentheses. My problem to date has been that Smalltalk seems to place additional single quotes in the strings that I create.

Answer I don't think what you are trying to do is how I would go about it. Host variables are only of simple type. They cannot be a collection of things. It would make more sense if you were trying to pass this to the IN: (:hostVar, :hostVar ....). This, as you mentioned, does seem to add an extra set of single quotes around each string.

So, with that said, I would recommend doing it by actually using a dynamic Where clause. There is an example on how to use one in the *User's Guide* on page 201. This allows you to build the Where clause via script. It seems as though you would need to pass your collection to the script and build the Where clause dynamically with this collection.

Errors when Binding to Database

Question I am experiencing a problem creating a DB2 query from VisualAge for Smalltalk, Version 2.0. I have tried the following procedure. First I issued the following commands from an OS/2 command line :

SQL

```
SQLBIND ABTD3220.BND SAMPLE /F=ISO SQLWARN NO
SQLBIND ABTD1620.BND SAMPLE /F=ISO SQLWARN NO
```

Second, I connected to the DB2 database from the Transcript window. Finally I dropped the query part onto the free-form area and opened the settings to create a query. When selecting **Insert** from the options, I received an error message saying that the SQLDA had set the SQLVAR to 0 and that the DB2 DBMS required at least 2 in this field. Did I overlook anything?

Answer To get VisualAge for Smalltalk, Version 2.0, to run with DB2/2 2.1, you need to rebind to your database from within the DB/2 command line processor using an additional parameter:

Try doing a connect to your desired database from CAE/2 OS/2 DB2 command processor:

- CONNECT TO databasename

- BIND ABTD3220.BND DATETIME ISO SQLWARN NO
- BIND ABTD1620.BND DATETIME ISO SQLWARN NO

It is important that you do the connect and the bind from within the command line processor.

Automatically Connect and Log-on to Database

Question When we test our application, the system prompts us for a Data Source Name every time we access the application. Would it be possible to write a piece of code that automatically does a connect and log-on to an ODBC database?

Answer Here is some Smalltalk code that can be used to automatically connect and log-on to an ODBC database:

Smalltalk

```
| conSpec logonSpec |
conSpec := AbtDatabaseConnectionSpec
  forDbmClass: #AbtOdbcDatabaseManager
  dataSourceName: 'FoxProC:'.
logonSpec := AbtDatabaseLogonSpec
  id: ''
  password: ''
  server: 'FoxProC:'.
conSpec connectUsingAlias: 'FoxProAliasC'
  logonSpec: logonSpec.
```

SQL Support

Question Does VisualAge for Smalltalk support the construction of any SQL statement? For example, can I issue these commands from an SQL editor?

- SELECT FOR UPDATE
- FETCH
- LOCK / UNLOCK
- COMMIT

Answer The SQL Editor does not support construction of the SELECT FOR UPDATE statement; however, you can use the Manual edit option to construct such a query.

The fetch operation is actually handled by using a VisualAge for Smalltalk AbtResultTable instance. Manipulation of the resultTable is hidden from the user when using the VisualAge database query parts; however, you can write a Smalltalk script to perform this type of manipulation (see example below).

Also, lock/unlock statements can be built using the Manual create option of the Database Query.

The Smalltalk script below was created without the visual query parts. It executes successfully against an Oracle database connection, but should be useful for IBM and ODBC as well:

Smalltalk

```
| connection querySpec table rt oldRow newRow |

"V3 protocol -> connection := AbtDbmSystem activeDatabaseConnection."
connection := AbtDbmSystem activeDatabaseMgr databaseInUse.
table := connection openTableNamed: 'ACCOUNTs'.

connection executeSQLStatement:
   'LOCK TABLE ACCOUNTS IN SHARE MODE'.

querySpec := 'SELECT * FROM ACCOUNTS WHERE ACCOUNT_ID',
  ' = 1  FOR UPDATE OF BAL'.
"Executing the above query causes the result set to be locked"

rt := connection resultTableFromQuerySpec: querySpec.

"#next message performs a cursor 'fetch' operation"
oldRow := rt next.
newRow := oldRow deepCopy.
 newRow at: 'BAL' put: 135.

"Statement below causes a standard 'UPDATE' statement to be
executed"
table atRow: oldRow putRow: newRow.

"VisualAge IBM database support has a positioned update feature.
See AbtIbmResultTable.  Update the row where the cursor is currently
positioned.  For example:  rt atCurrentRowPutRow: newRow."

"Close the resultTable.  This causes the cursor to be closed"
rt close.
connection commitUnitOfWork.

"Alternatively, you can execute the commit as follows"
connection executeSQLStatement: 'COMMIT'."
```

Using One Database with Different Database Managers

Question We want to develop an application that should use a database that resides in different database managers. We intend to exploit the native access to DB2 and Oracle and use the ODBC support for the other situations. We are working with VisualAge for Smalltalk, Version 3. I would like to have an answer to the following questions:

1. How should I design my application in order to reuse my data logic for the different database managers?
2. Would it be possible to create one application that supports all of the database managers I intend to use? For example, would VisualAge for Smalltalk give you the possibility of creating different paths depending on the database manager you want to use?

Answer

1. By changing your database connection specifications, you can cause the same database logic to be executed against different database sources. There is a limitation: When the VisualAge for Smalltalk SQL Editor saves queries containing host variables, the host variable representation is saved in a format that is tied to the database connection used when building the query.

   Queries that use host variables can be saved without information about the variable characteristics, but these queries will perform slower. Use the ′Manual create′ option to build queries that do not retain information about their host variable shapes.
2. It seems possible to use subapplications that contain information specific to the database manager you are working with.

You would:

- Define a subapplication for each of the database managers to be supported
- Create configuration expressions for each of the subapplications, so that only the applicable subapplication gets loaded, based on the state of the image (for example, if ODBC loaded, load ODBC subapplication)
- Create an access set (with the same name) in each of the subapplications. Only one of the subapplications can be loaded at any given time if you do this.
- Define your database connection specification in the subapplication.
- Define your queries (with **no** host variable information) in an access set found in the application. Alternatively, you could define your queries in

the subapplication, but you would have to build and maintain the query in three different places.

- Verify proper prerequisites for each of the application line-ups (for example, each subapplication has different prerequisites)

Query Not Found in Database Access Set

Question After I save a query (no matter which type) in a nonvisual part and restart VisualAge for Smalltalk, get the following error: ABT.SQL.9.w. It says that the query is not found in the database access set. However, if I look in the proper access set I can see that the query is specified. The application that uses these ″not found″ queries works fine if you edit the queries. Why do I have to edit them to make my application work?

Answer Below is some information which should prove useful in helping you locate the edit time representation of your query. When VisualAge saves a query specification, it actually saves two types of information.

1. A run-time selector is saved in the access set to build the query at application execution time.
2. An edit-time object is saved in the repository (manager). In addition to the run-time representation of the query specification, the edit-time object contains information to reconstruct the edit-time settings in the SQL Editor.

When looking for a particular query specification, the Database query settings view recognizes only existing queries for which both of the above pieces of information exist. Since you′ve verified that the run-time selector exists, it appears that VisualAge cannot find the edit-time representation of the query. If you are using the VisualAge for Smalltalk Professional (Team) development environment, check for other editions of the access set (use the ′Editions′ option from the VisualAge organizer).

Be aware that loading the run-time selector into your image will allow the query to execute, but the edit-time representation (stored in the manager) is associated with a particular edition of the access set. You can inspect the following line of code to view the edit-time query specification names in your access set:

Smalltalk

```
< access set > querySpecNames
```

Using getQuerySpecNamed:

Question I am developing an application that uses DB2/2 support. I was very pleased with the method *getQuerySpecNamed:* It allowed me to define query specification visually and use it in a script. However, I was quite surprised to see that this method is considered to be an edit-time method, since it is defined in the abtEditDatabaseSupport application. Is there a reason for this? Am I wrong in using this method in a run-time script?

Answer You should not be using this method at run time. This method is an internal method and is only used at edit time. Not every class and method supplied with VisualAge for Smalltalk is part of the API. It is very important to understand that VisualAge for Smalltalk includes not only the classes and methods that compose the API, but also the classes and methods used in the internal functioning of the VisualAge for Smalltalk APIs and the VisualAge for Smalltalk environment and its tools. The non-API classes and methods are primarily provided to aid developers in understanding the functioning of the system and in debugging their usage of the API.

Setting MaximumNumberRows

Question How can I set the query part's *maximumNumberRows:* attribute?

Answer Listed below is a Smalltalk query specification example that sets the maximumNumberRows to 5:

Smalltalk

```
| activeDatabase querySpec result resultCollection |
resultCollection := OrderedCollection new.
activeDatabase := AbtDbmSystem activeDatabaseMgr
  openDatabaseNamed: 'ORDERENT'.
querySpec := (AbtQuerySpec new)
  statement: 'SELECT CUSTOMERS.CUSTOMER_NUMBER,
    CUSTOMERS.NAME FROM CUSTOMERS'.
result := activeDatabase resultTableFromQuerySpec: querySpec.
result maximumNumberRows: 5.
result do: [:row |
  resultCollection add: (row asString)].
^resultCollection.
```

Accessing a Database Using Smalltalk

Tip

To access a database using only Smalltalk, begin with modeling. Do some analysis and design of your business and decide what object models your business needs.

For example, if you are querying a customer database, I would have a customer object that contains data particular to a customer. Then, I would have, possibly, a customer list. This customer list would be an ordered collection of customers.

Here is an example of code that builts a result table in Smalltalk and stuffs it into a table part of the window. However, the table part must already have its columns set up corresponding to the database columns. In other words, the attribute names for each column must be the database column names (in upper case). The example is:

Smalltalk

```
getResults

| dbmgr db table querySpec rt rc |

dbmgr := AbtDbmSystem activeDatabaseMgr.
db := dbmgr openDatabaseNamed: 'CLIENTLG'.
table := db openTableNamed: 'USERID.CLIENT'.
querySpec := (AbtQuerySpec new)
  statement: 'select * from CLIENT' .
rt := db resultTableFromQuerySpec: querySpec .

rc := (( AbtQueryResultTable new )
  querySpec: querySpec ;
  database: db ;
  resultTable: rt; yourself ).

(self subpartNamed: 'Table') rows: (rc rows).
```

The following example of code takes a result table on the layout surface and puts the rows into an ordered collection. At the end of the code, it takes the new *theClientList* and stuffs it into a variable on the layout surface called clientCollection so that it is accessible by other parts that use this part:

Smalltalk

```
createAllClientList
"Perform the createAllClientList action."

| newClient theClientList theResultTableRows |
theResultTableRows :=
  (self subpartNamed: #'resultTable of DatabaseQuery')
    valueOfAttributeNamed: #rows selector: #'IS_rows'.
theClientList := OrderedCollection new.

(theResultTableRows) do: [:aRow |
  newClient := Client new.
  newClient myDatabaseRow: aRow.
  theClientList add: newClient].

(self subpartNamed: #clientCollection) value: theClientList.
self currentClient:
  (self subpartNamed: #'currentRow of resultTable of DatabaseQuery')
     value.
```

Providing User Feedback when Updating DB2

Tip

You can pause a process by using the Delay class, but it seems like something you shouldn't have to do. The typical way of coding this is as follows:

1. Put up progress indicator
2. Fork background process (forkAt: Processor userBackgroundPriority)
3. Background process calls "CwAppContext default asyncExecInUI: <update progress indicator>" occasionally.
4. Background process finishes, calls "CwAppContext default syncExecInUI: <close process indicator>".

Sometimes developers forget step (2) and run off the user interface process. It doesn't hang VisualAge for Smalltalk because the database component forks work processes, but they don't get user interface updates as they expect. The alternative approaches to background processing is covered in the *IBM Smalltalk Programmer's Reference*.

Executing a Stored Procedure

Question Is it possible to execute a stored procedure with VisualAge for Smalltalk, Version 2, via ODBC?

Answer VisualAge for Smalltalk, Version 2, ODBC does not support executing stored procedures visually, but you could create a Smalltalk script similar to the following:

Smalltalk

```
| activeDatabase querySpec result resultCollection |

resultCollection := OrderedCollection new.
activeDatabase := AbtDbmSystem activeDatabaseMgr databaseInUse.
querySpec := (AbtQuerySpec new) statement: '{call sp_who}'.
result := activeDatabase resultTableFromQuerySpec: querySpec.
result do: [:row | resultCollection add: (row asString)].
^resultCollection
```

This will return the active users on SQL Server as strings. The ODBC stored procedure calling convention is {call xxxx}.

Executing a Stored Procedure with Parameter

Question How can I execute a stored procedure call to DB2/2 with parameters using ODBC Windows CAE?

Answer Here is an example for calling a stored procedure that passes one parameter:

```
Smalltalk

| aDict querySpec ct |
aDict := Dictionary new.
ct := CompoundType new
  addField: ( AbtOdbcVarCharField new
    name: 'COL1';
    count: ( 20 )).
aDict at: 'COL1' put: 'CHUCKTEST'.
querySpec := (AbtQuerySpec new)
  statement: '{call STOREDPROC(?)}'.
querySpec hostVarsShape: ct.
AbtDbmSystem activeDatabaseMgr
  executeQuerySpec: querySpec
  withValues: aDict.
```

ODBC Keyword Limitation

Question It appears that VisualAge for Smalltalk limits the connection string keywords that it passes on to an ODBC driver to DSN, user ID and password. Are there other driver-specific specific keywords that can be sent in from VisualAge for Smalltalk?

Answer Most of the connection information can be entered in the ODBC.INI file under the headings of the specified drivers. Refer to the *ODBC Drivers Reference* information. The only information that can dynamically be sent in at run time is the user ID, password, and DSN.

High-Level Qualifiers

Question I need to change the high-level qualifier in my application. Is there a way to do that without having to go into each query and modify it? Is it safe to edit the DB_allSpecs method and change from there?

Answer There is a way to change a high-level qualifier for all database access sets in an application for run time.

In order to create a database query, you need to specify a database access set. The database access set is a Smalltalk class that contains information about your queries. In particular, the private class method DB_allSpecs contains this information. You can manually change this method to specify

the new high level qualifier, or you can modify and run the code below to change the method.

If you created your database query using the SQL Editor (nonmanual option), the SQL Editor will not show this change. Consequently, if you update your query at a later time, DB_allSpecs will be generated with the old high-level qualifier. Remember to modify the method or run the code below again.

Once you've changed the high-level qualifier in the DB_allSpecs method, you need to look for the new high-level qualifier at development time (test option on the Composition Editor) and run time.

The best approach is to have one version of the class for development and one for run time (if you have VisualAge for Smalltalk Team). Make sure to release the correct version to your application before packaging it for run time, or create a development and a run-time application with the appropriate class. The code is as follows:

Smalltalk

```
 ″this code will change a high level qualifier for all
 database access sets in an application″

 | app oldHLQ uOldHLQ newHLQ uNewHLQ aSet uStmt oStmt
 aChanges sChanges start index |
 app := yourApp.            ″set this to the app you want altered″
 oldHLQ := ′oldHLQ′.        ″set this to the old high level qualifier″
 newHLQ := ′newHLQ′.        ″set this to the new high level qualifier″
 uOldHLQ := oldHLQ asUppercase.
 uNewHLQ := newHLQ asUppercase.
 ( AbtDbmSystem accessSetNamesForApp: app )
   do: [:aClassName |
     aChanges := false.
     Transcript cr;
       show: ( ′Examining Access Set - ′,aClassName ).
     ( aSet := aClassName abrAsClass ) registeredQuerySpecs
       do: [:aQuerySpec |
         oStmt := aQuerySpec statement.
         uStmt := oStmt asUppercase.
         sChanges := false.
         start := 1.

 ″for readability the next whileTrue^ loop is shifted to the left″

  [(index := uStmt indexOfSubCollection: uOldHLQ starting start ) ~= 0
]
  whileTrue: [
    oStmt := ( oStmt copyFrom: 1 to: index - newHLQ),
      (oStmt copyFrom: index + oldHLQ size to: oStmt size).
    sChanges := true .
    uStmt := oStmt asUppercase.
    start := index + newHLQ size. ].

    (sChanges) ifTrue: [
          aQuerySpec statement: oStmt.
          aSet putQuerySpec: aQuerySpec.
          Transcript cr; show:
            ( ′Changes made to Query Spec - ′, aQuerySpec name )
        ].
       ]. ″ end query specs for access set ″
    ]. ″end access sets ″
```

SQL Insert in Plain Smalltalk

Question Is it possible to perform a SQL Insert statement using plain Smalltalk? If so, how do I do this with multiple rows of data?

Answer The following code examples will insert into the database; you can add looping to add multiple rows:

Smalltalk

```
"Example - insertFromScript"
| db dbManager dict qSpec queryStatement |
dbManager := AbtDbmSystem activeDatabaseMgr.
db := AbtDbmSystem activeDatabaseMgr
  openDatabaseNamed: 'COMPANY'.
queryStatement := 'INSERT INTO EMPLOYEE
  (EMPLOYEENUMBER, NAME,DEPTNUMBER,DESCRIPTION)
  VALUES (:EENUM,
    :EMPLOYEENAME,:DEPTNUM,:DESCRIPTION)'.

qSpec := AbtQuerySpec new
  statement: queryStatement;
  yourself. dict := Dictionary new.
dict at: 'EENUM' put: '267267'.
dict at: 'EMPLOYEENAME' put: 'Joe Tex'.
dict at: 'DEPTNUM' put: 'TK5B'.
dict at: 'DESCRIPTION' put: 'Sales Rep'.
db executeQuerySpec: qSpec
  withValues: dict.
```

Smalltalk

```
"Example insert statement"
| dbm db qs sel |
db := AbtDbmSystem activeDatabaseMgr
  openDatabaseNamed: 'CLIENTLG'.
dbm := AbtDbmSystem activeDatabaseMgr.
sel := 'INSERT INTO CLIENT
  (NAME, COMPANY_NAME, PHONE_NUMBER)
  VALUES(''Mr. Grinch'', ''Mean&Heartless'', ''123-4567'')'.
dbm beginUnitOfWork.
qs := (AbtQuerySpec new) statement: sel.
dbm executeQuerySpec: qs.
dbm commitUnitOfWork.
```

Creating a Table in Smalltalk Code

Question How can I create a Table in DB2/2 using Smalltalk code?

Answer You can create a table in Smalltalk code by executing the following code (Make sure to connect to the database first.):

Smalltalk

```
| sqlDef dbmgr db table |
dbmgr := AbtDbmSystem activeDatabaseMgr.
db := dbmgr openDatabaseNamed: 'DBNAME'.
sqlDef := '(COL1 date  not null,',
  ' COL2 varchar(20), ' ,
  ' COL3  int,',
  ' COL4   decimal (6,2),',
  ' COL5 char(1) ,',
  ' COL6 varchar(10))'.
table := db
  createTableNamed: 'TABLENAME'
  definition: sqlDef.
```

Specifying Host Variables for a Query

Question How can I manually set host variables for a query?

Answer The following example selects rows from a table using an access set and host variables:

Smalltalk

```
| activeDatabase result resultCollection querySpec varDict |
querySpec := (YourAccessSet
   getQuerySpecNamed: 'showColumnNames').

resultCollection := OrderedCollection new.
activeDatabase := AbtDbmSystem activeDatabaseMgr
 openDatabaseNamed: 'YourDatabaseName'.

"Here you need to create a dictionary in which you
associate each host variable name with its value"

varDict := Dictionary new.
varDict at: 'tableName' put: 'theActualValue'.

"Then you use the resultTableFromQuerySpec:withValues: message "

result := activeDatabase
 resultTableFromQuerySpec: querySpec
 withValues: varDict.

"This would also work without an access set
 querySpec := (AbtQuerySpec new)
   statement: 'SELECT name FROM sysibm.syscolumns
   WHERE tbname = :tableName '.  ..."

result do: [:row |
 resultCollection add: (row asString)].
^resultCollection.
```

Deleting Rows from Database

Question I am trying to delete a number of rows from a database (DB2/2 Version 2.1). I think of using the following approach. First, declare a cursor to get the rows and second, skip through the rows and delete those that I don't want. Would this be the right approach?

Answer The idea will work. It is advisable to use public API methods when building your scripts. In this case, the query result table is the recommended mechanism. The cursor classes contain no public protocol and are not intended for direct use by programmers. The cursor is indirectly manipulated via the protocol defined in the AbtIbmResultTable class. To accomplish the intent of your script, you might consider using code like that shown below.

Just for your information, to browse the API methods for the database component, select the **Browse category** option from the **Smalltalk tools** option of the Transcript. Database run-time APIs are categorized as AbtDbRun-API.

Smalltalk

```
| connection qspec table rt oldRow newRow |
connection := AbtDbmSystem
  activeDatabaseConnectionWithAlias: 'VAIBMSamples'.
qspec := 'SELECT * FROM NameOfTable FOR UPDATE'
  abrAsQuerySpec.

rt := connection resultTableFromQuerySpec: qspec.
oldRow := rt next.
[ rt atEnd ] whileFalse: [
  rt deletePresentRow.
  rt next
].

rt close.
connection commitUnitOfWork.
```

Another potential option is to use the DELETE statement to remove the rows without having to fetch them first:

```
Smalltalk

| connection qspec table rt oldRow newRow |
connection := AbtDbmSystem
  activeDatabaseConnectionWithAlias:  'VAIBMSamples'.

qspec := 'DELETE FROM TableName WHERE F1 > 20'
  abrAsQuerySpec.
connection  executeQuerySpec: qspec.
```

Searching for Database Connection Errors

Question What approach can I use to trace an error when trying to make a request to a database?

Answer A trace of the SQL operations performed by your application would be useful. Below are some instructions to help you capture the trace information:

- Disconnect your active IBM database connection.
- Establish a new connection to the target database.

The above steps ensure a clean start.

- Execute the following command to start tracing of database calls:

```
Smalltalk

AbtDbmSystem activeDatabaseMgr startTraceOn: 'trace.out'.
```

- After the error message is returned, execute the following command to stop tracing of database calls:

```
Smalltalk

AbtDbmSystem activeDatabaseMgr stopTrace.
```

- Scan the trace file TRACE.OUT for anything that looks unusual.

Disabling Error Message

Question I want to handle the AbtError resulting from a query. I don′t want the error message to be displayed to the user. I can do this by subclassing the classes under AbtDatabaseQuery and overwrite the *defaultErrorBlock* method, but I would like to know if there is a switch to disable this option in all the queries.

Answer One way to overwrite the *defaultErrorBlock* is as follows:

- Extend the AbtDatabasePart class in your application
- Add an *initialize* instance method setting the (individual) errorBlock to whatever you want, making sure you have *super initialize* as the first statement.

Windows 95 and DB2

Question We have been working with VisualAge for Smalltalk Professional for OS/2, with heavy use of the IBM Database option (DB2/2 and MVS DB2). We now have some Windows 95 clients. Does Windows 95 only allow for access to DB2 via the call-level interface and using ODBC rather than native DB2? Is native DB2 access not possible?

Answer DB2 support on Windows 95 and NT is through the DB2 CLI. The CLI is native support. It is possible to use the CLI as an ODBC driver, but that is not what is done. The support is coded directly to the CLI. Furthermore, the future direction will be to use the CLI on all platforms. There is information about this in the database readme file, including information on how to migrate.

Database Parts for Windows 95

Question We are porting our VisualAge for Smalltalk, Version 3.0a, applications from OS/2 (professional client with server) to a Windows 95 client. Our applications use the database query parts off the parts palette (multirow query and the old query part). Since the Windows 95 version of VisualAge for Smalltalk uses IBM′s CLI for DB2, does anything have to be changed? Should we be able to do our queries without any changes?

Answer When porting to the call-level interface, it is necessary to migrate the host variable shapes that are saved in your query parts. An application is provided, AbtEditDatabaseMigrationApp to assist you. Load the application and execute *AbtDatabaseMigrationView new openWidget*. There should be some guidance on how to use it in the database readme.

Windows 95 and native DB2 DLLs

Question We were using the IBM database option in the OS/2 version of VisualAge for Smalltalk Version 3.0a, Professional. The class AbtIbmDatabaseManager implements a message called #sqlca, #sqlstate and #sqlcode (although they are private methods). We intensively make use of these methods in our application.

On switching to the Windows 95 platform, we cannot use the native DB2 DLLs. Instead, we must use the AbtIbmCliDatabaseManager class. This class does not implement the #sqlca, #sqlstate, and #sqlcode messages. Why were they not implemented, and is there a way to get this information?

Answer The AbtIbmCliDatabaseConnection stores the last error to occur in a dictionary accessed by #prevError. The dictionary key is the SQL state, and the value is a string that includes the SQL code and the associated message. The sqlcode is part of the string that is stored in the prevError dictionary. You can write a method to pull the sqlcode out of the string. The CLI did not have a way to retrieve an SQL communication area (SQLCA) until Version 2.1 of DB2/2, so if you have that version you will be using a subclass of AbtIbmCliDatabaseConnection, namely AbtIbmCliCSDatabaseConnection. This class has a method, #getSqlcaForStatementHandle:ifError:, which can be used to retrieve the sqlca. This method uses the #buildSqlca method to create an empty SQLCA to pass along to the CLI. The statement handle is saved in the cursor object (AbtIbmCliCursor). Each result table has a cursor. The selector is #hstmt. Be aware that this is a private method that might not be supported in future releases.

Scrollable Cursors

Question I am working on an application that searches a database for information. It will be used in an environment where a telephone operator answers questions online. Hence an extremely quick response time is needed. It won't be necessary to get more than 10 records.

I have written my own persistence layer methods and I decided not to use visual parts for database access. To accomplish the above, I was thinking to implement and manage scrollable cursors, however, I welcome any other suggestion.

Answer If you are using result tables (AbtResultTable, or subclass) in your persistence layer, you could do the following. Use #for:do:ifError::, the first argument is the number of rows to retrieve. For example, (rt is an AbtResultTable rows is a collection):

Smalltalk

```
rt
  for: 10
  do: [:row | rows add: row]
  ifError: [:err | err display].
```

The #for:do:ifError: does not fetch all rows. It fetches up to the number specified in the first argument. It leaves the cursor open, so the next time you use it, it picks up where it left off, and fetches the next n rows. There are other methods, like #do:ifError:, that fetch all rows, but #for:do:ifError: does not.

Database Connection Information

Question In the IBM database manager, there is a message called #connectionInfo. This allows you to retrieve the connection user ID. Is there some kind of similar method in the IBM call-level interface database manager?

Answer The message #connectionInfo was private, so it wasn't added to the CLI implementation. To get the user ID, add the following method to AbtIbmCliDatabaseConnection:

Smalltalk

```
userName
  "Answer the user name."

  | infoBuffer |
  infoBuffer := ByteArray new: 9.
  self
    getInfo: 47 "SQL_USER_NAME"
    buffer: infoBuffer
    ifError: self errorBlock.
  ^infoBuffer asString abrTrimNullsFromEnd
```

Moving from ODBC to CLI

Question How could we move from an application using ODBC to access DB2/6000 to an application that uses CLI in VisualAge for Smalltalk Windows 95, Version 3.0a? We have only one access set and think we could do the changes by hand. However, we are not convinced that this is the right way to go. We welcome any other idea or suggestion.

Answer The CLI database manager is a subclass of the ODBC database manager. The CLI database manager's type dictionary consists of all the ODBC data types and those data types unique to IBM CLI, such as large object (LOB) fields. Therefore, there is no need to change your queries, only your connection specification needs changing. Do the following to migrate your application to IBM CLI from ODBC:

1. Catalog the DB2/6000 database you'll be accessing.
2. Update your connection specification, changing the database manager to IBM Database 2 - CLI.
3. Set the database source name to the alias you used to catalog the DB2/6000 database.

To take a look at the types supported by a CLI database manager, inspect the following code snippet after connecting to a CLI database (replace 'alias' with your connection alias):

```
Smalltalk

(AbtDbmSystem activeDatabaseConnectionWithAlias: 'alias')
  databaseMgr class typeDict
```

Undefined Access Set

Question I have tried to port an DB2/2 application from Version 3.0 of VisualAge for Smalltalk to Version 3.0a. Testing the application revealed that I had lost my database connection. When I tried to recreate it, I was told I had not yet defined an access set. However, my access set class is present within the application. Any ideas on what is happening?

Answer VisualAge will attempt to build a connection specification only when an access set is actually defined within an application.

An access set that is extended by an application will not be recognized when the database connections view attempts to create a new connection specification. In order to edit or create a connection specification, you must select the application where your access set is defined, then open the Database connections view.

To determine which application owns the access set, display the result of the following statement:

```
Smalltalk

<MyAccessSet> controller
```

The connection specification is merely a method in your access set. If the above advice is not helpful, you can:

- Double-click on your access set to open a class browser.
- Manually create a new class method (like that below) that matches the name of your connection alias.
- Add the method to category 'Connection specs' (the database query parts search for defined connection specifications based on category name).

Smalltalk

```
"Sample of connection specification method for Oracle"

 ^AbtDatabaseConnectionSpec new
   dbmClass: AbtOracleDatabaseManager;
   dsn: 'Current Oracle Database';
   promptEnabled: true;
   yourself.
```

The Database connections view displays information about connection specifications defined by an application; however, the database query parts (on the palette) have access to all connection specifications that are visible to the application (all connection specifications defined within the application or one of its prerequisites).

Table and View Names

Tip The queries to retrieve table and view names from a database are hard-coded in the public methods of AbtIbmDatabaseConnection. Methods with prefix 'all*' are typically used to gather lists of tables or views, for example, *allUserTableNamesIfError:* (there are several others also). You can modify the queries in these methods to point at the views that exist for your installation.

Use of the Multirow Query Settings

Question How do I use the multirow query settings?

Answer Use it to set the maximum number of rows to fetch (used with a one-time query to limit the rows returned Enable packeting) for DB2/2, quick forms, and containers. It returns up to the number of rows that will fit on the display area. You can get more rows as needed to satisfy user actions, such as scrolling, and enable blocked fetches. For ODBC, you can up to the number of rows that you specify in the Blocked fetch size for each request

DBF Format

Question Does VisualAge for Smalltalk support DBF format?

Answer Yes. VisualAge for Smalltalk supports ODBC and there are ODBC drivers for dBase files.

ODBC Support for OS/2

Question A hypothetical company wants to develop applications for laptops 12 MB of RAM and prefers not to use DB2/2 because of its license costs and memory requirements. Is it cheaper to use an ODBC-driver for dBase files or something similar? What about the Multidatabase Feature that is bundled with the base product (single user and team versions) and should support ODBC. Are all drivers mentioned in the ODBC document a part of VisualAge for Smalltalk? Are users allowed to deploy these drivers without charge?

Answer If you are using the Explore CD, the answer is that you can't distribute it and it would not run if you did. If you bought VisualAge for Smalltalk, then you can distribute the ODBC DLLs that came with the product without charge. (The ODBC support came with the base product.) If you had the Explore CD and ODBC support did not come with it, then you need to get a new version of the Explore CD that comes with the ODBC support.

Object-Oriented Databases

Question Does VisualAge for Smalltalk work with OO databases?

Answer Versant and GemStone are the ones that work today. GemStone has a very good object-oriented DBMS for use with Smalltalk, including VisualAge and IBM Smalltalk.

Absence of Database Query Fields

Question What would cause my database query to show no fields?

Answer One of two things:

1. The database is not set up or connected to properly. See the *VisualAge for Smalltalk User's Guide* for connecting to a database. If you are using DB2, make sure you also bind to the database. Use your DBMS tools to set up the database schema.
2. The query is not set up properly. The database query part has a settings popup. In the resulting dialog, define your query. For more information, refer to the *VisualAge for Smalltalk User's Guide*.

Support for Blocking with Oracle

Question Does VisualAge for Smalltalk have support for blocking with Oracle?

Answer For Native Oracle, yes. For ODBC no (although it may use it under the covers).

Comparing ODBC and Native Oracle Interfaces

Question How does the ODBC interface compare with a native Oracle database interface?

Answer Performance differences between ODBC and native Oracle are not extremely significant. Native Oracle has ″blocked fetching″ capability, and performance can be tuned by the programmer by specifying an optimal number of rows to fetch. However, we suspect that the ODBC driver is also taking advantage of Oracle′s blocked fetching capability (under the covers) since data retrieval speeds are quite comparable. Both ODBC and native Oracle have the ability to INSERT/UPDATE multiple rows with a single database call. Both ODBC and native Oracle provide a stored procedure interface; however, the ODBC interface does not support passing array parameters to stored procedures (the native Oracle interfaces supports array parameters). Native Oracle has definite advantages where application setup is concerned. ODBC requires the configuration of an .INI file for each machine where the application will run, and you must ship support DLLs with your packaged application. Native Oracle uses the OCI DLL supplied with Oracle; therefore, no additional files or setup is required when bundling the run-time application. The configuration issue becomes even more significant when deploying an application on multiple platforms. These are really the only differences.

SQL0805N Message when Creating a Database Query

Question I get a SQL0805N message ″NULLID.ABTD3200 is not found. SQLSTATE= 51002″ when attempting to create a query on a new database. What am I doing wrong?

Answer This message signals that you forgot to bind VisualAge to your database. For IBM Database Managers, you must bind to a database before using it. If you are using VisualAge for Smalltalk for OS/2, you can do this from the Transcript window by typing the following where ′dbname′ is the name of the database you are binding to:

Smalltalk

```
AbtDbmSystem activeDatabaseMgr
  bindToDatabaseNamed: 'dbname'
```

If you prefer, you can bind to the database from an OS/2 command line:

```
sqlbind abtd3200.bnd 'dbname'
sqlbind abtd1600.bnd 'dbname'
```

If you are using VisualAge for Smalltalk for Windows, you must first create an empty database from your OS/2 server machine. Then type the following command at an OS/2 command prompt:

```
sqlbind abtdw20.bnd dbname /F=ISO
```

For more information on the Sqlbind command, consult the *DB2/2 Command Reference* manual.

Checking Multidatabase Feature Installation

Question I want to check to see if we have already installed the Multidatabase Feature of VisualAge for Smalltalk. How do I proceed?

Answer To determine if you have previously installed the Multidatabase Feature, follow these steps:

1. Start VisualAge on your workstation.
2. On the System Transcript window, select **Smalltalk tools** → **Browse Application**.

3. In the dialog that appears, type *AbtDbmMultiDbBaseApp* and then select the **OK** push button.

   If the Application Browser opens on AbtDbmMultiDbBase application,″ then you installed the Multidatabase Feature previously, and you should follow the instructions for ″Multidatabase Customers.″ If, instead, a message box displays with the message ″AbtDbmMultiDbBaseApp is not the name of an application,″ then you have not installed the Multidatabase Feature and you should follow the instructions for ″ODBC Customers.″

4. After determining whether or not, you have previously installed the Multidatabase Feature, exit VisualAge for Smalltalk.

   If you already have the Multidatabase Feature installed, be sure to follow the instructions for installing ODBC titled ″Multidatabase Customers.″

Changing the Database Name in All Application Classes

Question Before I package my application to send it to my customer, I need to change the database name in all my database query parts. Is there an easy way to do this without opening each part ?

Answer Execute the following in your Transcript window or in a workspace. Make sure to change MyApp, OLDDB, and NEWDB. Ignore the message written to your Transcript.

Smalltalk

```
| record bldr |

MyApp allLocalClasses do: [ :class |
 ((class inheritsFrom: AbtAppBldrPart)
  and: [(bldr := (record := class partBuilderRecord) builder) notNil])
   ifTrue: [bldr subpartBuilders do: [ :subbldr |
    (subbldr attributeSettingNamed: #databaseName) = 'OLDDB'
    ifTrue: [ subbldr attributeSettingNamed: #databaseName
        put:'NEWDB' ] ].
 class partBuilderRecord: record ] ].
```

Nature of ODBC

Question What is ODBC?

Answer The IBM VisualAge for Smalltalk ODBC Support provides support for an Open Database Connectivity (ODBC) interface between VisualAge for Smalltalk applications and databases. The ODBC interface enables you to create VisualAge database applications that are independent of a specific database management system. ODBC Support provides a single ODBC interface between your VisualAge applications and the specific database drivers that you use to access databases.

Problem in Specifying Driver

Question When I was connecting to my database, I get the message ″Driver specified by data source name could not be loaded (4149).″ What's wrong?

Answer This problem can happen when connecting to an Oracle, SQL Server, or Sybase database.

During installation, you should have been asked whether or not you wanted to initialize the odbc.ini file when you installed the VisualAge for Smalltalk SQL Server or Oracle support. If you did not answer *yes,* type and execute the following in your Transcript for Sybase or SQL Server:

```
Smalltalk
AbtMultiDbSqlSvrDatabaseManager installIniFile
```

or, for Oracle,

```
Smalltalk
AbtMultiDbOracleDatabaseManager installIniFile
```

Other causes of this problem may be that the location of DLLs starting with ″qe″ are not specified in the libpath in the config.sys (or path in the autoexec.bat on Windows). These qe*.dll files are supplied with VisualAge and reside in the \VISUALAG\DLL directory on the client machine (standard VisualAge for Smalltalk) or server (VisualAge for Smalltalk Team). For Sybase or SQL Server, the required file is qess04.dll and for Oracle,

qe704.dll. File qelib.dll is also an important DLL, required by both databases.

Other potential causes of the problem using Oracle are that the ORA7032.DLL file required by the VisualAge for Smalltalk code is not present or its location is not specified in the libpath in the config.sys (or path in the autoexec.bat on Windows). Oracle users must type a ′b:′ before the server name on the connect if using SQL*Net for NetBIOS or ′t:′ for TCP/IP.

Other potential causes of the problem using Sybase or SQL Server are that the SYBDB.DLL file required by the VisualAge for Smalltalk code is not present or its location is not specified in the libpath in the config.sys (or path in the autoexec.bat on Windows).

For Sybase or SQL Server, Versions above 4.6, or for Sybase System 10, the user must install client code from Sybase that supports older versions of Sybase (has the sybdb.dll file, not the newer libsybdb.dll file).

Also, users should remember that the server name must be typed in the correct case (lower or upper) that was used during installation. In order to check this, the user may look at the sql.ini file in the SQL Server ini subdirectory. The ini file must be edited with a ini editor. However, you can still see the server name just by typing the sql.ini file.

User ID Not Privileged for Read

Question I get a message SQL0551N; user ID does not have the privilege to perform read. What′s wrong?

Answer Make sure first that the user you are logged on to has authorizations to use the table. Second, issue authority to the user ID that performed the bind on the database. This is for VisualAge for Smalltalk, Version 2.0:

```
SQL
Grant run on package nullid.abtd3220 to userid        or
Grant run on package nullid.abtd3220 to public
```

Error: Oracle ORA-00942

Question I receive an error message — 'Oracle ORA-00942: table or view does not exist' — when clicking on a table name when using the query builder in VisualAge for Smalltalk. Why?

Answer Oracle 6 had an entry in the all_synonyms table for accessible_columns. That entry pointed the synonym accessible_columns to the table all_tab_columns. Oracle 7 does not include this table, which VisualAge for Smalltalk needs. To get this table, you can either create a synonym in the all_synonyms table of accessible_columns and point it to the table all_tab_columns and then VisualAge for Smalltalk should run okay.

Or, Oracle 7 provides a script named CATALOG6.SQL. This script creates Version 6 views for compatibility. The script resides in the RDBMS70\ADMIN subdirectory. If you run this script (you probably need DBA authority), it will create the ACCESSIBLE_COLUMNS synonym that VisualAge for Smalltalk uses. For details on this script, see the *ORACLE7 Server Migration Guide.*

Changing a High-Level Qualifier for Run Time

Question The high-level qualifier I use at development is different from the one I use at run time. How can I change it for run time ?

Answer In order to create a database query, you need to specify a database access set. The database access set is a Smalltalk class that contains information about your queries. In particular, the private class method DB_allSpecs contains this information. You can manually change this method to specify the new high-level qualifier, or you can modify and run the code below to change the method.

If you created your database query using the SQL Editor (nonmanual option), the SQL Editor will not show this change. Consequently, if you update your query at a later time, we will generate DB_allSpecs with the old high-level qualifier. Remember to modify the method or run the code below again.

Once you've changed the high-level qualifier in the DB_allSpecs method, we will look for the new high-level qualifier at development time (test option on the Composition Editor) and run time. The best approach is to have one version of the class for development and one for run time (if you have

VisualAge for Smalltalk, Team). Make sure to release the correct version to your application before packaging it for run time or create yourself a development and a run-time application with the appropriate class.

Smalltalk

```
" this code will change a high level qualifier for all
  database access sets in an application "

| app oldHLQ uOldHLQ newHLQ uNewHLQ aSet uStmt oStmt
  aChanges sChanges start index |

app := yourApp.          "set this to the app you want altered "
oldHLQ := 'oldHLQ'.      "set this to the old high level qualifier"
newHLQ := 'newHLQ'.      "set this to the new high level qualifier"
uOldHLQ := oldHLQ asUppercase.
uNewHLQ := newHLQ asUppercase.
( AbtDbmSystem accessSetNamesForApp: app )
    do: [:aClassName| aChanges := false.
      Transcript cr; show: ( 'Examining Access Set - ',aClassName ).
      ( aSet := aClassName abrAsClass ) registeredQuerySpecs
      do: [:aQuerySpec| oStmt := aQuerySpec statement.
          uStmt := oStmt asUppercase.
          sChanges := false.
          start := 1.
          [( index := uStmt indexOfSubCollection: uOldHLQ startingAt:
                start ) = 0 ]
            whileTrue: [ oStmt := ( oStmt copyFrom: 1 to: index - 1),
              newHLQ,
              ( oStmt copyFrom: index + oldHLQ size to: oStmt size).
              sChanges := true .
              uStmt := oStmt asUppercase.
              start := index + newHLQ size.
              ].
          ( sChanges )
            ifTrue: [ aQuerySpec statement: oStmt.
              aSet putQuerySpec: aQuerySpec.
              Transcript cr; show:
                  ( 'Changes made to Query Spec - '
                    ,aQuerySpec name ) ].
      ]. " end query specs for access set "
    ]. "end access sets "
```

DB2 Cursor Control

Question How do I perform DB2 cursor control in VisualAge for Smalltalk?

Answer VisualAge for Smalltalk does not currently support the control of cursors for DB2. However, this level of control can be done with Smalltalk coding.

Below are some ideas for manipulating SQL cursors for IBM database queries. Many implementation details are left out, but this information should give you a starting point for enhancing the database visual part.

Cursor manipulation is possible using the AbtResultTable class. AbtResultTable objects are required for any query that returns data to VisualAge for Smalltalk, such as SELECT statements.

Below is a snippet of code that demonstrates a subset of the AbtResultTable function. The #next message is equivalent to a cursor fetch. The #close message closes the cursor.

Smalltalk

```
| dbm db rt row |
dbm := AbtDbmSystem activeDatabaseMgr.
db := dbm openDatabaseNamed: 'VISUALAG'.
rt := db resultTableFromQuerySpec:
('SELECT * FROM SQLCAM01' abrAsQuerySpec).
[ (row := rt next) == nil]
 whileFalse: [ Transcript show: row asString; cr ].
rt close.
```

One way to override the result table (cursor) logic is to modify the Database Visual part code. This code resides in application AbtRunDatabaseQueryPartApp.

Create a subclass of AbtQueryResultTable class so that certain methods can be overridden. Method #initializeOnResultTable is used to build a collection of rows from the result table. The default implementation is to load ***all*** rows into the collection using the AbtResultTable method do:. This method can be overridden to initially retrieve only a certain number of rows.

```Smalltalk
(( self resultTable ) == nil )
ifFalse:[self getMoreRows: 10].
"getMoreRows: must be written"

self currentRowIndex: (( self rows size > 0 )
   ifTrue: [ 1 ]
   ifFalse: [ 0  ].
```

The *getMoreRows:* method must be created, and it will be used to get the specified number of rows. The application developer will be responsible for closing the cursor when finished.

```Smalltalk
getMoreRows: aNum

 | newRow rowCount |
 rowCount := 1.
 newRow :=  self resultTable next.
   [newRow == nil or:[rowCount > aNum ] ]
    whileFalse: [
    self rows add:
    ( self updateObjectForRow: newRow alreadyInDb: true ).
    rowCount := rowCount + 1.
    newRow := self resultTable next. ]
```

Using the Public Interface editor, the user can create an action in the newly created class to retrieve the next n rows.

AbtDatabaseQuery methods *executeQuery* and *executeQueryAsTransaction* contain code to construct an AbtQueryResultTable object based on the query being executed.

These methods would need to be overridden to instantiate the new AbtQueryResultTable subclass (rather than AbtQueryResultTable).

Database Feature not Installable

Question I would like to use the Database feature of VisualAge for Smalltalk, but it is not on my install selection. I am trying out the OS/2 VisualAge for Smalltalk Team from the Explore CD. I need to access a Database but am unable to install a database manager.

Selecting 'Install...' from the **Smalltalk Tools** menu in the System Transcript provides only selections for DynamicDataExchange, Smalltalk Programming Examples, SOMsupport and TrailBlazer. I have installed all available server options (including IBM Database and C).

When I try to install the Multidatabase feature (I want to access an Oracle v7 database) there is no extra selection to install. What did I overlook?

Answer VisualAge for Smalltalk features require a two-step install. Be sure that you have installed the IBM Database feature from the CD, then check your install selections from the Smalltalk **Tools** pull-down menu. Also, check the install path in the abt.cnf to see if is pointing to the correct path. The abt.cnf file resides in the \visualag subdirectory on the client machine.

If your install path in the abt.cnf seems correct, try executing the following from within the Transcript window:

Smalltalk

```
System setSubsystemType: 'ABT' to: 'VA'
```

Once this is executed, the ability to install the multidatabase, C, and COBOL features should be on the install menu.

Database Support for Windows

Question What kind of database support is there for VisualAge for Windows?

Answer Database support includes many databases. Included with the base product is support for IBM DB2/2. Previously, support for accessing Oracle or SQL Server was available through the Multi- database feature for VisualAge for Smalltalk. Recently this was replaced with our new ODBC support, which is free to users who have the base product. With ODBC support, many additional databases are now supported.

In order to take advantage of DB2/2 under Windows, customers must purchase DB2/2 or DB2/6000. Using a product called DB2 Client Application Enabler/DOS (DB2 CAE/DOS), Windows clients can access either DB2/2 or DB2/6000. DB2 CAE/DOS and other DB2 CAE products are offered free of charge. With the help of a product called DDCS/2, clients may access DB2 data o MVS and AS/400.

Formatting rowsAsStrings to Remove Brackets

Question How do I format the string returned from *rowsAsStrings* from the database result table to get rid of the brackets and other funny characters?

Answer If you would like to display just one column of the database table in the visual part (such as list box, combo box) then you can specify an attribute name in the settings for that visual part that is the same as the name of one of the columns in your database table. Then, connect the resultTable(rows) to the visual part(items).

The *rowsAsStrings* message puts the brackets around each column in the result row. The way to get around it is to pick out the data you want using an expression like:

Smalltalk

```
name := row at: #COLNAME  (Can do this if using Smalltalk
                                and not VisualAge)
```

If you would like to display more than just one column of the database table, you can use the following examples as reference. The *Multimedia User's Guide and Reference*, shows you in the insurance Sample application, how to write an attribute-to-script connection that formats the rows into nicer looking strings. This is the example in that manual

Smalltalk

```
formatNameList
  "Formats the names in the result table of the query as
     Lastname, Firstname, Middlename."

  | list oldList aString table |

  list := OrderedCollection new.
  table := (self subpartAttributeValue: #(#NameTable self)).
  table isNil ifTrue: [ ^list ].

  oldList := table rows.
  (oldList isNil) ifFalse: [
      oldList do: [ :row |
        aString := ( ( row at: #LASTNAME), ', ', (row at: #FIRSTNAME)
          ', ', (row at: #MIDDLENAME)
        list add: aString]
  ].

  ^list.
```

Another alternative is to add an attribute to the public interface of a part. This attribute can have a get method that returns the columns in a displayable manner somewhat like the above examples. Then specify that attribute name in the settings for your nonvisual part.

This is another example that fills a container from a query that retrieves client names from a database:

```Smalltalk
fillContainer
  ″fills the container with client icons″
  | client cnr icon rows name id |
  icon := (AbtIconDescriptor new
     moduleName: 'MTBMP10';
     id: 806;
     yourself).
  cnr := self partAttributeValue: #(#Container #self).
  rows := self partAttributeValue:
      #(#'resultTable of DatabaseQuery' #rows).
  rows do: [ :row |
   name := ( (row at: #LASTNAME), ', ',(row at: #FIRSTNAME),' ',
       (row at: #MIDDLENAME) ).
   id := (row at: #ID).
   client := AbtIconGadgetView new.
   client parentView: cnr;
    label: name;
    largeIcon: icon;
    openWidget.
   client abtWhen: #defaultAction
    perform: (DirectedMessage new
     receiver: self;
     selector: #openClient:id:;
     arguments: (Array with: name with: id)).
  ].
  cnr arrangeItems.
```

Intercepting a 100 or 0 SQLCODE

Question How can I intercept a 100 or 0 SQLCODE?

Answer To check that a query successfully retrieved data, you can check that the ″rows″ collection size is greater than 0. For example:

```Smalltalk
  (self rows size = 0)
      ifTrue: [ ...″SQLCODE = 100 logic″ ]
      ifFalse: [ ... ″SQLCODE = 0 logic″ ]
```

Database Access Set

Question What is a database access set and how do I create one ?

Answer A database access set is a VisualAge for Smalltalk class used for storing database query specifications and stored procedure definitions. It is created when you want to build a database query part.

As long as you use only one database manager, you can use the same access set for storing all of the queries and stored procedures in an application. An access set can be shared by different applications. If you use more than one database manager, however, create an access set for each database manager.

To create a database access set:

1. Select Browse Application on a new or existing application.
2. In the Application Browser, select **Create Database Access Set** from the Classes pull-down menu.
3. Type a name for your access set in the Database Access Set Name window, then select the **OK** push button.

Stored Procedures for SQL Server

Question How do I handle stored procedures for the SQL Server in VisualAge for Smalltalk?

Answer There are two implementations that will support SQL Server stored procedures in VisualAge for Smalltalk.

The first involves changing the VisualAge code. This will enable the customer to still use the visuals (the database query part, not the stored procedure part) much as you currently do a database query. This implementation also requires that the customer package his application as a run-time image or, if using the merge method, dump the changes to the VisualAge code to merge with his application.

This change can be overwritten with maintenance supplied by the VisualAge for Smalltalk support team.

The second implementation is totally programmatical and code is provided below under ″Implementation 2.″ This code must be written for every stored

procedure you wish to call. Alternatively, the customer can design a nonvisual reusable part along these lines.

Implementation 1:

Go to the **System Transcript** menu and click on **Smalltalk Tools/System/Change User** and change to Library Supervisor. Then, go to the **System Transcript/Smalltalk Tools** menu and click on **Manage Applications**. Find and click on **AbtRunDatabaseQueryPartApp**, and choose **Applications/Create New Edition**. Then, in the second pane, click on **AbtDatabaseQuery** and choose **Classes/Create New Edition**.

Drop a database query part on the layout surface of any composition editor. Bring up the context menu (second mouse button) and click on **edit part**. Then, go to the Public Interface Editor for that part, click on the actions page and add a new action name called *executeStoredProcedure.* In the action selector box, type *executeStoredProcedure*. Go to the script editor for that part, click on the **AbtDbRun-Internal category** and write a script called ″executeStoredProcedure″ that looks like the following:

Smalltalk

```
executeStoredProcedure
 ″Execute my query spec; no commits or rollbacks are issued.″

 | result |

 result := self executeQuerySetUp.
 (( result isAbtError ) or: [result == nil])
  ifFalse: [ result := self executeSelect ].
 ^result
```

Save the part, then go back to the Application Manager window, find **AbtRunDatabaseQueryPartApp**, and click on **AbtDatabaseQuery** in the second pane. Under **Classes** click on **Version Release All** and name the version something that you will recognize later.

Under **Applications**, choose **Version Applications/Name Each** and name the version the same as you did for the class.

Then, go to the **System Transcript** menu and click on **Smalltalk Tools/System/Change User** and change back to your user ID. In order for others in your group to take advantage of these changes, they will have to load this version of AbtRunDatabaseQueryPartApp in their image.

With this implementation, you can use the visual query part to manually create a query specification to call the stored procedure. Then tear off a result table from the query part. In order to present your data in a table format, you will need to drop a table part on the window and add columns to the table for each column that the stored procedure returns. For each column, you will need to open the settings and set the attribute name to the exact column name of the table in the database.

Then, make two connections from the table to the result table part. Connect the table (rows) to the result table (rows) and the table (selectedRowIndex) to the result table (currentRow Index).

Then, connect your event to run the query to the executeStoredProcedure action of the database query part.

If you package your application as a run-time image, then you will not have to perform the next step.

If you use the merge (.app) file packaging method, you will need to dump these changes and include them in your run-time application. To do this, execute the code below in a workspace or transcript window.

Smalltalk

```
|result|
(result := ApplicationDumper new)
  dump: (AbtRunDatabaseQueryPartApp)
  intoFileNamed: 'abtdbqry.app'.

result hasErrorOccurred ifTrue: [
   self error: 'An error occurred when dumping the application'.
     result currentErrorString ].
```

This will create a file in the current directory. This file must then be moved to your merge directory during the packaging step (*VisualAge for Smalltalk User's Guide*). The .CNF file needs to be modified and "abtdbqry.app" needs to be added directly after the database .app files in the list.

Implementation 2:

This code, when implemented in a script for a view, will run a stored procedure and place the results in a list box on the window called "List."

Smalltalk

```
 executeStoredProc
 " This code executes a SQL Server system stored procedure and puts
the
    result table in a list box on the window. "

 | dbmgr resultTable qSpec db |

  dbmgr := AbtDbmSystem activeDatabaseMgr.
  (dbmgr notNil)
   ifTrue: [
     db := dbmgr openDatabaseNamed: 'pubs'.
     qSpec := AbtQuerySpec new statement: 'sp_helpdb pubs'.
    resultTable := dbmgr databaseInUse resultTableFromQuerySpec:
qSpec.

    (self subpartNamed: 'List') items: (resultTable asStrings)
    ].
```

Error: SQL0236W

Question When I try to connect to my database, I get SQL0236W error. What is wrong?

Answer To get VisualAge Version 2.0 to run with DB2/2 2.1, you need to rebind to your database from within the DB/2 command-line processor using an additional parameter:

Try doing a connect to your desired database from CAE/2 OS/2 DB2 command processor:

- CONNECT TO databasename
- BIND ABTD3220.BND DATETIME ISO SQLWARN NO
- BIND ABTD1620.BND DATETIME ISO SQLWARN NO

Please note that VisualAge for Smalltalk, Version 2.0, does ***not*** allow you to take advantage of the new enhancements DB2/2 2.1 provides. These enhancements are supported only in VisualAge for Smalltalk Version 3 or higher..

Communication with DB2/2

Question I need to know all about the platforms that DB2/2 will run on and how VisualAge can communicate with DB2/2. Can you help me?

Answer IBM DB2 is a family of client-server RDBMS. As with any client-server RDBMS, clients and servers can and do work on different operating systems. DB2 servers, the products that actually manage data, are available for OS/2, RS/6000, HP, SUN, AS/400, VM, VSE, and MVS platforms. These are the platforms where you store your data and a particular DB2 server manages an orderly multiuser access to this data.

DB2 data stored on the server can be accessed from a variety of applications running on different client platforms. Visualiser and VisualAge for Smalltalk are an example of such applications. DB2 Family supports applications running on DOS, Windows, and OS/2 PCs, several flavors of UNIX (AIX, HP-UX, Solaris) as well as AS/400, VM, VSE, and MVS applications. At this time, some combinations of clients and servers because they need Version 2 of DB2 on OS/2 and UNIX. One example of a combination that does not work in time is an MVS application accessing DB2/2 data. One the other hand, a very popular combination that ***does work*** is Windows applications, such as Excel, WordPerfect, accessing DB2/MVS data.

DB2 provides application enabling to Windows products such as VisualAge for Windows and Visualizer for Windows (when they are delivered) so that they can access DB2 data no matter which DB2 server it resides on (MVS, O/2 and so on). The product that provides this enablement is called DB2 Client Application Enabler/DOS. (DB2 CAE/DOS). DB2 CAE/DOS, and other DB2 CAE products are offered free of charge.

Access to host databases requires use of a DDCS gateway. DDCS gateway products provide access to host data from desktop applications. The DDCS gateway itself is available for OS/2, AIX, HP-UX, and Solaris environments. However, it does support all of the DB2 CAE client products. In other words, DOS and Windows are supported as well.

Sample Databases

Question How do I install the sample databases that come with VisualAge?

Answer Before installing the sample database, make sure the IBM Database Feature is installed. To do this:

- Select Install from the **Smalltalk Tools** pull-down menu in the Transcript window.
- Select IBM Database and IBM Database Samples from the list and select **OK**.
- If using VisualAge for Smalltalk for Windows, make sure the following statement is in your AUTOEXEC.BAT file: SET DB2CODEPAGE=437

Next, be sure to activate a database manager. From the Transcript window, select Database from the Visual Tools pull-down menu. The Database Selection window is displayed:

- From the Database Manager drop-down list, select the database manager you want to use.
- If the particular database manager you are using requires a user ID, password, and server, enter the appropriate values.
- If you want, you may enter a preferred database in the text entry field. The SQL Editor uses this as your default database.
- Select the **Connect** push button.

To install the ORDERENT sample database, execute

```
Smalltalk
AbtDatabaseSamples installDatabase
```

in your Transcript window. See the *VisualAge for Smalltalk User's Guide and Reference* for more information.

For the multimedia database, execute

```
Smalltalk
AbtMultimediaSamplesApp installDatabase
```

More information about this can be found in the online Multimedia Guide.

ODBC Driver Error

Question I am getting a dialog telling me that I don't have a license for the ODBC Drivers. What do I do?

Answer Make sure that the QEIB.LIC is in the same directory as the drivers (IB***0*.DLL).

- For OS/2, the licensing file is called QEIB.LIC.
- For Windows, the licensing file is called QEIBM.LIC.
- If you have purchased drivers from Intersolv or Q+E before, your licensing file may be called something else or stored somewhere else.

Data Source Name Error

Question I am getting the message: "Data Source Name not found and no default name specified." What's wrong?

Answer Your ODBC.INI file does not contain the Data Source Name entry. Use the ODBC Administrator to add the entry (ODBCADM.EXE).

Minimum Files for Run-Time

Question What are the minimum files that need to be included at run time?

Answer

- Windows
  - IBUTL06.DLL
  - IBBAS06.DLL
  - ODBC.DLL
  - ODBCINST.DLL
  - IB***06.DLL ← Your database driver
  - ABTDWW20.DLL
- OS/2
  - IBUTL04.DLL
  - IBBAS04.DLL

- IBMDS04.DLL
- ODBC.DLL
- ODBCINST.DLL
- IB***04.DLL ← Your database driver
- ABTDPW20.DLL

In addition, for any flat file drivers you will also need:

- IBFLT06.DLL for Windows
- IBFLT04.DLL for OS/2

If you are using Non-IBM supplied drivers, you will need only:

- ODBC.DLL
- ODBCINST.DLL
- ABTDPW20.DLL (OS/2)
- ABTDWW20.DLL (Windows)

Data Source Driver Error

Question I am getting the message ″Driver specified by Data Source name could not be loaded.″

Answer Make sure the ODBC.INI entry for the specified Data Source Name is pointing to the correct driver.

Also, make sure the driver DLL can load. To verify this, use QEDLLOAD.EXE or CHKDLL32.EXE.

Database Access Set

Question Can I simply copy a private method, in which a query is held, of the DatabaseAccess class and save it with another name and then use this new method to set the query for a Single-Row Query part?

Answer VisualAge for Smalltalk stores hidden edit-time information about query specifications in the library. Unfortunately, merely making a copy of a method does not propagate the necessary edit-time information required by the database parts on the palette. The example below demonstrates how one might copy the same query to a different name. A similar technique could be used to copy specifications between different access sets.

Smalltalk

```
| sourceAccessSet targetAccessSet qSpec |
sourceAccessSet := targetAccessSet := MyAccessSet.
qSpec := sourceAccessSet getQuerySpecNamed: 'Query1'.
qSpec name: 'NewQuery1'.
targetAccessSet putQuerySpec: qSpec.
```

Using Stored Procedures with ODBC or DB2 CLI

Tip Unified field AbtDatabaseVarCharField does not map to the proper DBM-specific field at run time. Therefore, if you use stored procedures with ODBC or DB2 CLI, the text fields may be truncated on the left.

Active Database Connections When Exiting VisualAge

Tip Exiting VisualAge does not force a disconnect of an active database connection in VisualAge. Therefore, before exiting VisualAge, close all active connections to databases.

Unloading the Static SQL Feature

Tip If you have the Static SQL feature loaded, execute the following code and then unload the feature.

Smalltalk

```
AbtSqlIbmOs2Precompiler removing
```

Database Samples

Tip The general database samples can be packaged; however, they cannot be run as a packaged application.

Chapter 6. Distributed

In this chapter, we cover the VisualAge for Smalltalk Distributed feature.

Distributed Feature versus CORBA

Question I'm surprised by the fact that VisualAge for Smalltalk Distributed is not CORBA-compliant. Will this change in the future?

Answer The intent of the Distributed Feature is not to provide CORBA bindings. That is provided by the SOM/DSOM bindings in VisualAge for Smalltalk. The two do not overlap. They are both in the current version of VisualAge for Smalltalk and can both be used in the same image concurrently.

For good advice on what should be the preferable choice in a given environment (SOM/DSOM vs. Distributed Feature), consult *Developing Distributed Object Applications with VisualAge Smalltalk Distributed Feature*. Here are some points to watch for:

- Pure Smalltalk programming vs. language independence
- Storage management
- Transparency during developmant
- Communication protocols
- Distributed tools

Turning on Distributed Tracing

Question How do I turn on distributed tracing?

Answer To turn distributed tracing on, do one of the following:

- Start the image with the command line parameter -Dt. This will generate a file called ″dstrace.log″ that traces every step of the distributed operation.

  Note

  You have to add another packaging rule for this to work at run time. Include the global variable DsCommunicationNLS.

- Evaluate

```
DsTracer traceAllToFile: 'dstrace.log'.
```
or
```
DsTracer traceOffAll
```

Note

This feature is our internal debugging tool, and is unsupported

Avoiding Hard-Coding TCP/IP Addresses for Distribution

Question Is there a way not to hard-code TCP/IP addresses when using matrix-based distribution?

Answer The distribution matrix uses location references, which are stored in the name server. The location references contain the physical addresses of the machines on the network. To change the physical location of any location in the matrix, all you have to do is edit the name server. No change to the packaged image is necessary, assuming the name server is persistent. To make a name server persistent, use the **Archive** → **Make Persistent** pop-up menu in the Name Server Browser. In this case, the Name Server contents is stored into a file which is loaded on image start up. The name server information can be modified by editing or replacing the name server file.

Premature Connection Closure

Question What happens when the connection closes when I'm sending a message to another object space?

Answer Communication errors can be handled with exception processing. The general exception that covers distributed errors is:

Smalltalk

```
DsDistributedSystem unavailableObjectException
```

There are examples in the distrib.txt file shipped with the feature.

There are also callbacks for certain events, which can be registered for. The events include connections, disconnections, sends out/in, receives out/in, and so on. See class DsCallbackRec under the class category DS-EventCallbacks for the available callbacks. You can register for callbacks with

Smalltalk

```
DsDistributedSystem localSpace addCallback: callbackId
  receiver: anObject1 selector: aSymbol clientData: anObject2.
```

There is a complementary #removeCallback... method. See the cardgame demo, DtCardGameApplication for sample usage. The callback methods work the same way that widget callbacks work.

An important point to understand is that we recognize communications errors only when a socket call fails. When two object spaces are connected, each side has a socket-receive call pending. When the cables are disconnected, some TCP stacks will return an error on the receive, others will not. If you need more timely information on disconnects, I would suggest creating a watchdog process that sends periodic ping messages to check out the health of the connection.

When the two disconnected object spaces are reconnected, we attempt to recover to the state of all remote objects. If the object spaces do not match, they will be reset to an initial connection state. A common reason that the object spaces would not match, would be if the client object space needed to be restored from a saved image.

Remote Object Has No Object ID

Question I can′t start VisualAge for Smalltalk Distributed correctly. Smalltalk Distributed Initialization runs okay. VisualAge for Smalltalk Organizer does not show up and also can not be started manually.

Transcript window shows: ″CwAppContext: walkback during proc dispatch... all procs removed.″

The distributed debugger window shows ″ABT.DSF.35.i: Remote Object has no object id, DsParentLogicalProcess:Thread1-...suspended″. The only workaround is to use a fresh image.

Answer Apparently someone is sending #isClass to a remote object pointer during image startup. This usually occurs because there was a remote browser open when you saved the image or a remote object pointer was saved in the Smalltalk dictionary. If you have a classes browser open, it will send messages to every object in the Smalltalk dictionary when the window gets restored, looking for all the classes in the image. In the debugger that

shows up, try to find the global that has a remote object pointer in it, set it to nil, drop to the stack frame that restores the snapshot and resume the process.

Copying Objects Between Object Spaces

Question I'm trying to send objects by value rather than by reference. To do that, I'm using #asDsByValueObject. But the performance doesn't appear satisfactory. What should I do?

Answer The following text can be filed into your image to fix this problem:

Smalltalk

```
DsObjectMobility becomeDefault!

! DsInputStream privateMethods !

reconcileCopy: anArrayOfArraysOfInstances
  | beforeList afterList localSpace |

  beforeList := OrderedCollection new.
  afterList := OrderedCollection new.
  localSpace := objectSpace localSpace.
  anArrayOfArraysOfInstances do: [:instArray |
   instArray do: [:inst |
  inst dsBasicClass reconcile: inst
      afterCopyInto: localSpace
      before: beforeList
      after: afterList
  ]
  ].
  beforeList isEmpty ifFalse: [
    beforeList asArray multiBecome: afterList asArray]
!  !
```

Activator Wasting CPU Time

Question When VisualAge for Smalltalk is running without the activator, it consumes little CPU time when idle. But whan I run it with the activator, it consumes a considerable amount of CPU time even being idle. What's going on?

Answer Because Smalltalk implements a light process model, it has an idle loop that checks if timers have expired, This loop wakes up every 100 ms and accounts for the small CPU load (say, 5%) you see. When the system is busy, the idle loop is not, so most of that 5% is then used for useful work. When you connect to the activator, a similar loop is created to look for data from the activator. Again this loop will handle useful work if there is work to do. There are private methods that affect the timing of the loop but we don't suggest that you change them because they will affect the message delays when the system is busy.

Security Error With Connect Request on Activation

Question I have a client object space sending a request to the AIX server object space. The activator daemon starts up the VisualAge for Smalltalk image and this is what I get in the Transcript window:

```
Smalltalk
ABT.DSF.301.i: Smalltalk Distributed initializing...
ABT.DSF.309.i: Object space activated;with key 'aixspace' at host'aixsrv'
ABT.DSF.304.i: Smalltalk Distributed initialization complete.
ABT.DSF.413.e: Client is not authorized to access this server.
Activation Error - ABT.DSF.339.e: Security error with connect request
  from object space at host 'amtechvage' with key ''.
```

Why do I get this?

Answer A common reason is not having abtdsusr.cnf and abtdsaut.cnf files in the current directory of each of the two object spaces.

Server Without User Interface

Tip Even if your server does not need a user interface, I suggest that you create a small application window. This window at least makes it easy to see that your server is running and gives you a way to stop it. There are ways to create a headless server but they are complicated and not extensively tested.

Packager Method Exclusion Remedy

Question How do I know if I need the #packagerInclude... methods?

Answer If you have methods that are called by a remote client only, then the packager will exclude those messages, and maybe even the classes. The fix to this is to use the #packagerInclude... method to identify these methods so they will be included in the packaged image. This is not too bad, since you have to think about the API to your distributed object anyway. This is a good place to document it.

Remember that the selectors returned by *packagerIncludeSelectors* are class or instance neutral. That is, any instance or class method in an included class is included in the image, whether directly referenced or not.

Many people do not like having to have the #packagerInclude... method with all the necessary methods, but that's the way the packager works today. Because our packager is so good at getting rid of things in the image, we have to appease it by listing the things that we really want to keep.

The easiest way is to go ahead and package your client code in your server. If image size is really critical, then you'll have to either include references to classes and methods in a selector that you know will be packaged, or use the #packagerIncludeSelectors method. All you have to include in #packagerIncludeSelectors method are the messages that you expect a remote client to send. This should be in your design somewhere as the advertised API of your server. You may even be able to generate this method based on method categories.

You can find more information about #packagerInclude... in the *IBM Smalltalk User's Guide*. Also, you may want to look at the Packaged Classes Browser, **Method** menu.

Name Server Distributed Part Difficulties

Question I find that the name server distributed part from TCP/IP category doesn't work for some connections. How can I fix it?

Answer Try this little fix:

Smalltalk

```
DtRunDistributedObject becomeDefault.
DtRunDistributedObject isVersion ifTrue:
[DtRunDistributedObject newEdition]!

!DsRemoteObjectPointer publicMethods !

performIn: anObject arguments: args

^Processor dsActiveLogicalProcess
    forward: (DsSendOperation
      objectSpace: objectSpace
      markerBlock: [ :op | ^op ]
      receiver: self
      selector: #performIn:arguments:
      arguments: (Array with: anObject
      with: (args asDsParameterArray)) asDsParameterArray)
! !
```

Error During Distributed Load

Question I'm trying to run an example from the *VisualAge for Smalltalk Distributed User's Guide*. (The example with the DtListBuilder application). After creating configuration map and distribution matrix, when I try to perform DistributedLoad, I get the next: ″...Abt.Dsf.1.i: Connected to object space with key ... at host ... ″, and then, suddenly : ″Abt.Dsf.709.e: Failed trying to load AVBMap ″, where AVBMap is the name I gave to the configuration map. What could cause this?

Answer You can't use the **Distribution Matrix** → **Distributed Load** menu item when your object spaces are connected to different ENVY libraries. What would it load? The configuration map is not in the other library.

Testing Distributed Smalltalk for TCP

Question What can I do to test Distributed Smalltalk for TCP/IP?

Answer Try these methods:

```
Smalltalk

DsDistributedSystem testTCPLocal
DsDistributedSystem testTCPAddress: 'xx.xx.xx.xx'
DsDistributedSystem testTCPHostName: 'host.domain.com'
```

You may also want try these:

```
Smalltalk

DsTCPCommunicator localHostAddress.
DsTCPCommunicator localHostName.
DsTCPCommunicator hostNameFromAddress: '1.2.3.4'.
DsTCPCommunicator fullHostNameFromName: 'hostname'.
DsTCPCommunicator hostAddressFromName: 'hostname.company.com'.
```

Distributed Initialization on Windows for Workgroups

Question When I try to load the Distributed Feature into my image, the code loads and a message displays in the Transcript that says "Smalltalk Distributed Initializing...," but it never finishes. I left it overnight and the next morning it was still running. I have verified this is not a problem with host-name resolution. I am running Windows for Workgroups, Version 3.11 and am using the Microsoft TCP/IP stack. I first loaded TCP/IP into my image and then the Distributed feature.

Answer This is a known problem with Microsoft Windows for Workgroups TCP/IP, Version 3.11b is a complete refresh and solves the problem you have. Have a look at *ftp://ftp.microsoft.com/bussys/Clients/WFW/update.txt*

Anyway, if you are running the MS Windows for Workgroups TCP/IP stack, make sure you are running Version 3.11b or later. See the above address for information on fixes and upgrades.

Seeing Your Data Moving

Tip After quite a bit of tuning, my application runs over a telephone line (28.8 KB). One advantage of using the modem is that I can *see* data moving (watching the lights) and it gives a good feel for when you think stuff is on the client, but you actually reference it on the server and so on.

Another way to ″See data moving″ is to open the Profiles Browser and create an object space profile (**Profiles** → **Remote Objects** → **Profile the Object Space**), then press **auto refresh**. Profiling the object space has a relatively small performance impact (compared to profiling each or all remote objects).

Role of the Activator

Question Can objects that reside in the server image send messages to objects that reside in the client image without having the activator running on the client machine?

Answer Sure they can. The Activator is involved only in establishing the initial connection between the two object spaces. After that, they communicate directly without any involvement by the activator. In this sense, the activator is somewhat like the *inetd* super-server in TCP/IP.

Once the connection is established, all communication is peer-to-peer. Don′t worry too much about the ″client/server″ distinction: the only difference between the two is that a server can receive incoming connections and a client cannot. However, once the connection is made, both sides can send and receive messages.

Remote Object Pointer Dead After Packaging

Question When I run my application in the distributed development environment, the local and remote object spaces communicate just fine. When I package my local image and run it, I get a run-time error saying “Remote object pointer is dead.” I′m running the run-time application from the same machine as the development environment, so I know my TCP/IP environment is OK. I′m running the remote development environment as the server, so that part should certainly work, since the local development image can talk to it (I did a Reset Distributed System to break the connection to the local development image).

Answer You will see a dead remote object pointer when a message is sent to a remote object that has been reset. Remote objects are reset when you do a DsDistributedSystem reset, or reset the connection. Also, when two object space are connected and their states do not match (images were saved at different times), then we automatically reset during the connection. To avoid the error, your application will need to clear the pointer at the appropriate time. There are callbacks available for connections, disconnections, and distributed system resets to aid you.

When you send a message to a shadow or a shadow-loaded class like #self, then what you get in return is not a shadow, but a lighter weight Remote Object Pointer. This object is as good as a proxy in that it will forward any messages to the real remote object but it can only reconnect to the object space where it was created. That object is time stamped and everything. The shadow however attempts the connection from scratch if it loses contact with the object space.

Remote objects that you reference through shadows are okay, because they contain the information to reconnect to the remote objects. The remote objects that you have to worry about are the intermediate objects that are returned as results from other message sends. For example, when you send a message to a shadow, it may return another remote object.

One thing that might help is to do a DsDistributedSystem reset prior to switching into packaging mode.

Logging Transcript Messages

Tip

At run time, you can log the Transcript messages to a file with the **-l** command line parameter, that is,

ABT.EXE -lfilename.log -iruntime.img

Tracing Run-Time Startup Problems

Tip

You can trace the run-time image startup sequence by putting Transcript show:'s in YourApplication class>>#startUp, EmSystemConfiguration>>#startUp, or even in EmSystemConfiguration>>#startup. At run time, you can log the Transcript messages to a file with the **-l** command line parameter.

Here is a short explanation of the run-time startup sequence which may help you choose the right trace points:

The first method loaded by the VM is EmSystemConfiguration>>#startup (note the lowercase **u**). After performing EmSystemConfiguration>>#initializeVM, this method creates a user interface process to execute EmSystemConfiguration#startUp. What goes on in #startUp is best explained by this comment in the source code:

Smalltalk

```
EmSystemConfiguration>>startUp

"Inform each loaded application and subApplication that the image
 is about to be started up. After the initialization of the vm the
 startup sequence is as follows.

1. Set ImageStarted to false.
2. Perform pre-startUps. Perform any work, which must be performed
   only before the standard startUp sequence.
3. Set the System #breakMessage: and send #initializeNlsSystem to the
   startUpClass.
4. Send #localize to all (Sub)Applications in the required order.
5. Send #initializeSystem to the startUpClass.
6. Send #startUp; #relocalize to all (Sub)Applications in the required
   order.
7. Set ImageStarted to true.
8. Send #run to the startUpClass."
```

Name Discrepancy Problem in Loading

Question I'm using a configuration map that contains two location references, client and server, and I'm using a distribution matrix with this map. When doing a distributed load, the locations in the map are loaded according to the matrix. There are two other developers on my team who need to start using the Distributed feature. When they try to do a distributed load of the config map, the server loads fine but the client fails because the other developers host names and IP addresses are different from what I defined in the client location reference. The only way I can think of to get around this is to have a different configuration map for each developer, but it would be difficult to keep them all synchronized.

Is there another way of handling this? I have shadow classes in my server that reference real classes in the client. Will the shadow classes in the server image use the client location reference and if so, doesn't that mean that my server object space will be able to interact with only one client object space?

Answer The design point of the Distribution Matrix was that it could be defined once and used by several developers. The only items the Distribution Matrix saves in the library are the application names and the location names. The location names are resolved (through the name server) in the individual developer's image at the time the distributed load is requested. Therefore, all the developers should share the same configuration map (with the Distribution Matrix) and should have location references with the same names in their name server. However, each can have different definitions of those location references.

As to having shadow classes in your server that reference real classes in the client: I think something is backward here. I'm not saying that it is impossible to have shadows in your server object space, but it is not a good idea to have them pointing to client object spaces. You don't know the address of your client ahead of time, and in most cases you don't even know how many clients you may have at any one time.

Shadows are almost always used in clients to point to classes in the server or to other well known remote objects.

Sharing Class Instances Across Object Spaces

Question When two clients make a connection to the same remote class, are they able to share all the instances of the class together?

Answer Yes. Distributed Smalltalk maintains the object identities across object spaces.

Peer-to-Peer or Server

Question Is the peer-to-peer object communications true peer-to-peer, or is it a set of communicating servers, like what happens in many CORBA implementations?

Answer Yes, it is peer-to-peer, in that once a connection is made, either side can initiate a message to the other side. True, in order to make the first connection one side has to act like a client and the other has to act like a server. But, I think it is better than a ″set of communicating servers″ because the side that played the server role first doesn't have to locate the client in order to send a message to it.

Equivalent of a Threads Package

Question Is there an equivalent of a threads package in distributed?

Answer VisualAge for Smalltalk Smalltalk has a built-in process model. You can create many independent Smalltalk processes. However, all these Smalltalk processes run on the same native process/thread.

In addition VisualAge provides a mechanism to perform a primitive call on an independent native OS process/thread.

Sending an Object as a Parameter to a Remote Object

Question What happens if an object is sent as a parameter to a remote object. Is it cloned, copied, reference-passed, or moved to the remote image?

Answer The default is to pass all parameters by reference. You can override this default on a parameter by parameter basis to make a copy. We do not yet have any built-in replication support, so you are on your own if you copy.

Sending an Object to a Remote Site While a Thread Executes

Question What happens if I attempt to pass an object to a remote site while a thread is executing in one of its methods?

Answer It depends on what you mean by ″pass an object.″ You can pass a reference to an object to a remote site at any time, regardless of whether one of its methods is currently executing. The only other operation we allow is to copy. The only consideration here is whether the state of the object is inconsistent at some point in the middle of the method. We do not yet support actually moving an object to another node, although you could build this on top of the existing support.

Disconnecting Remote Images

Question Is it possible for remote images to be disconnected without stalling everything?

Answer Yes. Your application can register and receive a callback when a connection to a remote object space is lost. It is up to your application to handle that in an appropriate way.

Dynamic Change Potential

Question How tightly coupled is the system? Can new clients be added on the fly? Can servers be modified on the fly?

Answer The system has a late binding in that you can use the Name Server to locate objects at run time. The information in the Name Server can be changed at any time, so any request for the named object will be directed to the new location.

Once the connection is made, it is tightly coupled, as long as you hold on to the remote object instance. You have to go back through the Name Server to take advantage of any real-time changes.

Clients can connect and disconnect from a server as much as necessary. I'm not sure what you mean by "servers be modified on the fly". Once a connection is made (a remote object located in a server object space) another server can't automatically take over that server's work. But if the client relocated the remote object through the Name Server, as stated above, the newly located object could reside in another server.

Retention of Instances

Question I am trying to perform a distributed load, as described in the *VisualAge for Smalltalk Distributed User's Guide*. On step 2 (see page 34 in the *Guide*) I receive an error message saying DtListBuilderCollection cannot be removed because it has instances. What do I do?

Answer Use the following code:

```
Smalltalk
DtListBuilderCollection allInstancesDo: [ :each | each become: nil ]
```

TCP/IP Errors at Startup

Question I get TCP/IP errors at Distributed Smalltalk startup? What do I do?

Answer Check to make sure that your TCP/IP name server is working correctly. You should be able to ping yourself and others, both by name and by IP address. If you do not have a name server, you will need to make an entry into the file TCPIP\ETC\HOSTS (your TCP/IP directory may be named differently). The TCP/IP also has a user interface that updates this file for you.

Distributing Parts

Question Is there any graphical mapping of the physical distributed environment in a way that will enable me to move parts of the application from one machine to another?

Answer We have an extension to the Configuration Map object that allows you to design which applications are to be loaded into which object spaces. This is called the *Distribution Matrix*. You define the locations (client or server roles or clusters) and it displays a matrix. You can use the matrix to specify which applications should be loaded on which locations. This information is used when you load your Configuration Map; it does not do an immediate move. The advantage is that this information is saved in the library (connected to the Configuration Map) to be used again by you or anyone else. Since the locations are resolved to real object spaces through the name server, this information is dynamic; it depends on the current mapping.

Tools to Manage Network Traffic

Question Are there any tools that can monitor and manage the flow of traffic in the network?

Answer One option in the Profiler (**Transcript** → **Distribution Tools** → **Browse Profiles**) monitors and profiles all messages in and out of a specific object space. We do not have any tools that cover the scope of more than one object space. There are also no tools that allow you to trap and modify any messages.

SOM/DSOM Implementation

Question Can a SOM/DSOM application be distributed?

Answer In VisualAge for Smalltalk, Version 3.0, you can communicate with a SOM/DSOM client object. We do not yet support creating SOM/DSOM server objects in the image. You can use the SOM/DSOM client support in any distributed object space. However, the Distributed feature cannot help implement SOM/DSOM or CORBA server objects.

Fault Tolerance for Object Spaces

Question What fault tolerance is available with object spaces?

Answer We handle all communication errors as Smalltalk exceptions. Your application code should catch these errors (when:do:) whenever you do any remote messaging.

Changing Name Server Entries at Run Time

Question Can name server entries be changed dynamically at run time?

Answer Yes, this is the reason the name server exists. If you reference an object through a name server object reference, then every time you retrieve that information you will use the current information in the name server. This is how shadows work.

Long Startup Delay

Question What might cause a long delay in starting up VisualAge for Smalltalk Distributed?

Answer That ugly delay is probably the result of the Distributed feature attempting to locate your host name in the name server. This would happen if you are not pointing to a name server and don't have your local host name in your /etc/hosts file.

Packaging an Application

Question How do I package a distributed VisualAge for Smalltalk application?

Answer The first step in packaging is to use the menu item **Distribution tools** → **Options** → **Enable Packaging**. While packaging is enabled, you cannot communicate with another object space. You then proceed much as you would with a regular VisualAge application: for simple applications, you can use the **Make Executable** menu choice. For more complex applications, you may have to use the **Browse Packaged Images** menu option at least once to specify your packaging rules. DtShadows act like regular objects during the packaging operation and are included in the run-time image to again forward messages.

Object Spaces from Different Libraries

Question Can object spaces run from different team server libraries and still use distributed load?

Answer Only if both libraries have the required code.

Message Tracing

Question Is there a way to trace the messaging in Distributed Smalltalk?

Answer Yes, but it causes a large volume of output so be careful. Evaluate the following in your client and server images, respectively:

Smalltalk

```
DsTracer traceApplicationToFile: 'client.trc'
DsTracer traceApplicationToFile: 'server.trc'
```

Then run your distributed application and look at the TRC files.

Handling TCP/IP Addresses

Question How are TCP/IP addresses handled in a packaged Distributed Smalltalk executable?

Answer The distribution matrix uses location references, which are stored in the name server. The location references contain the physical addresses of the machines on the network. To change the physical location of any location in the matrix, all you have to do is edit the name server. No change to the packaged image is necessary, assuming the name server is persistent.

Making the Name Server Persist

Question How do I make the name server persistent?

Answer The name server browser has a pop-up menu (**Archive** → **Make Persistent**) that allows you to make a name server persistent. In this case, its contents are stored in a file that is loaded on image start up. The name server information can be modified by editing or replacing the name server file.

Capturing Information from a Walkback Window

Question How can I capture information when I get a walkback window in a development environment?

Answer At the debugger window, select **Stack** then **Write Stack Trace Text To File**. This report provides messaging information to help debug the problem.

Copying Object Space to a Backup Processor

Question Is there an option to duplicate or copy an object space into another processor as a backup?

Answer We don′t have anything like this built in. You can use the capabilities that are in the product to design your own standby object space. However, you can′t redirect all pointers your application may have to any remote objects. Some of the remote object pointers are not relocatable. You would have to design your application so it can (1) keep a hot standby up to date with all the current state information; (2) detect when to switch to that standby; and (3) refresh any saved object that may be a remote object pointer. You could use the Name Server and Shadows to help.

Error: Client not Authorized for Server

Question What causes the error ″ABT.DSF.413.e: Client is not authorized to access this server?″

Answer You should have abtdsusr.cnf and abtdsaut.cnf files in the current directory of each of your client and server object spaces. Both of these files are installed on the client. You can execute ″DsSecurityConfiguration authorizedUsersFile″ and DsSecurityConfiguration passwordsFile″ to see the names of the two CNF files. You can locate them in different directories by explicitly stating the complete pathname in these CNF files. You can also control access by stating in these files exactly what object spaces have access rights (the default is ″* *″, which gives everyone access).

Error: Remote Object Has No ID

Question What causes the error ″ABT.DSF.35.i: Remote Object has no object id?″

Answer One cause results in a message #isClass to a Remote Object Pointer during image startup. This usually occurs because a remote browser was open when you saved the image, or a Remote Object Pointer was saved in the Smalltalk dictionary. If you have a classes browser open, it will send messages to every object in the Smalltalk dictionary when the window is restored, looking for all the classes in the image. If you can see the rest of the stack, try to find the global that has a Remote Object Pointer in it, set it to nil, drop to the stack frame that restores the snapshot, and resume the process.

Extending Server Classes

Question Can server classes be extended in client object spaces when using a distribution matrix?

Answer No. The way shadow load works is to replace the existing class with a shadow to the class that exists only in the server object space. Since the class no longer really exists in the client object space, you can′t extend it there. The class, and all instances of the class, exist only in the server object space, so you can extend the class only in the server object space. You can, however, have class extensions in subapplications and any other application that is actually loaded in the same place.

Moving Parameters Between Client and Server Object Spaces

Question What is the best way to get parameters between client and server object spaces?

Answer Assuming that you are sending the message from the client to aRemoteObject, you want to copy the result of the message, which may be another remote object. The best way to do this would be:

Smalltalk

```
(aRemoteObject message: parm)
    dsInSpace: (DsDistributedSystem localSpace).
```

You can also try

Smalltalk

```
object dsAsUnloadedBytes asString
```

before sending the object to the client. If you want to be sure that the parameter is also copied from the client to the server, so the server won′t have to send any remote messages, then you can either make sure the parameter is a string or integer (sent by value by default) or use the asDsByValueParameter message. To keep performance optimal, you may also want to convert your objects to strings first, before sending the message.

Improving Performance Across Client-Server Object Spaces

Question What can I do to improve performance across client-server object spaces?

Answer You can marshal your data. Marshaling is converting a real object into a String, binary, or other portable form. There are three ways to marshal remote objects:

- Shallow:

```
Smalltalk
myLocalCopy := myRemoteObject dsInSpace:
  (DsObjectSpace localSpace).
```

- Deep:

```
Smalltalk
myLocalCopy := myRemoteObject asByValueObject.
```

- Application:

```
Smalltalk
myLocalCopy := MyLocalCopyClass fromMarshalData:
  (myRemoteObject myMarshalingData).
```

The last technique is where you write your own marshaling.

Partitioning Objects Across Object Spaces

Question Are there any rules of thumb on how to partition objects across object spaces?

Answer Yes. Minimize cross-object-space message traffic. When possible, combine multiple messages into one message and make cross-object-space messages asynchronous. Design your application to be sliced so you can manage distributed messaging.

Transaction Management Provision

Question Is there transaction management in Distributed Smalltalk?

Answer No, we don't have a Transaction Service implemented in Distributed Smalltalk. We are looking into it and know it is a high priority. Until it exists, you can implement some concurrency control in your server objects with the use of semaphores.

Loading the Distributed Feature

Question What is the best way to load the Distributed feature?

Answer Load TrailBlazer, TCP/IP, and Distributed features, in that order. Then apply any known fixes.

Packaging an Application

Question What are the steps for packaging a Distributed Smalltalk application?

Answer

1. Create the configuration map.
2. Create the locations. Remember, these have to point to things in the name server.
3. Create the Distribution Matrix, and fill it in.
4. Load your configuration map locally to test it and make changes.
5. Do a Distributed Load of your configuration map at least once before packaging.
6. For each location/platform combination:
    a. Load the location into a fresh image, just to make sure no garbage is hanging around.
    b. Package using either the one button packager, the package option on the applications browser, or **Smalltalk tools** → **Browse Packaged Images**.

Allowing Clients Run-Time Access

Question What must be done to allow run-time access to distributed clients?

Answer You must copy abtdsaut.cnf and abtdsusr.cnf from your development image directory to your packaged image directory.

Retaining Needed Classes and Methods

Question What do I do when the packager strips out classes or methods I need in a client?

Answer The packager is good at getting rid of things in the image we don't need. You have to list the things that we really want to keep if they are not referenced elsewhere. The easiest way to do this is to go ahead and package your client code in your server. If the image size is really critical, then you'll have to either include references to classes and methods in a selector that you know will be packaged, or use the #packagerIncludeSelectors method. This is not really too bad, all you have to include in #packagerIncludeSelectors method are the messages that you expect a remote client to send. This should be in your design somewhere as the advertised API of your server. You might even be able to generate this method based on method categories.

Run-Time Startup Problems

Question What do I do if I have run-time startup problems when using Distributed Smalltalk?

Answer Make sure you have the following setup in your packaging. If you are packaging an application with a Shadow Load (see Chapter 6, "Distributing an Application with a Distribution Matrix," in the *VisualAge for Smalltalk Distributed User's Guide*) then you may need to change the prerequisites on the DtCodeDistribution application:

1. Remove DtCommon and LibraryObjects (leave DtNameServerApplication)
2. Add DsMRISupport.

One more thing you will have to change is the packaging rules. Add the following packaging rules to your application:

```
Smalltalk
  packagingRulesFor: aPackagedImage
  ″Define rules for the given packaged image.″

  aPackagedImage excludeVariable: ′deadROP′ inClassNamed:
#DtShadow.
  aPackagedImage excludeVariable: ′lastROP′ inClassNamed:
#DtShadow.
  aPackagedImage includeClassNamed:
#DtGlobalReferenceWithLocationReference.
```

Finding Remote Object Space Pointers in Image

Question What can I do to make sure I have no remote object space pointers in my image?

Answer If you know that you do not want any remote object pointers or object space connections saved in your image, you can execute:

```
Smalltalk
  DsDistributedSystem saveConnectionState: false.
```

If this option is set *false*, on start Up, your image will start clean. If you need this only for packaging, you can set up a packaging rule to set the DsActivation class variable ′SaveConnectionState′ to *false*.

Debugging a Client System

Question How can I debug a distributed client system?

Answer Three things to try, in order.

1. Is there a walkback.log file? If there is a problem, there should be one in your run-time directory.
2. Try starting your run-time application with the parameter ″-lruntime.log″. This will create a run-time.log file containing anything that might be sent to the Transcript plus any debugger messages.

3. If neither of the above gives you any information, then try our internal (unadvertised, unsupported) trace. Add the parameter -dt when starting your run-time image. This will generate a file called dstrace.log that traces every step of the distributed operation.

By the way, you have to add another packaging rule for this to work at run-time. Include the global variable DsCommunicationsNLS.

Fault Tolerance

Question How fault-tolerant is Distributed Smalltalk?

Answer The fault-tolerant behavior that you see is dependent on the TCP/IP Stack and the operating system. We recognize communications errors only when a socket call fails. When two object spaces are connected, each side has a socket receive call pending. When the cables are disconnected, some TCP stacks will return an error on the receive, others will not. TCP may not return an error until a send call is issued. Distributed Smalltalk will occasionally send control messages for things like Distributed Garbage Collection, which might detect a disconnect. If you need more timely information on disconnects, I would suggest creating a watchdog process that sends periodic ping messages to check out the health of the connection. When two disconnected object spaces are reconnected, we attempt to recover to the state of all remote objects. If the object spaces do not match, they will be reset to an initial connection state. A common reason that the object spaces would not match would be if the client object space needed to be restore d from a saved image.

If your application needs more control, there are several techniques that an application can use. Here is a brief summary: Communication errors can be handled with exception processing. The general exception that covers distributed errors is:

Smalltalk

```
DsDistributedSystem unavailableObjectException.
```

There are examples in the distrib.txt file shipped with the feature.

Using TCP/IP Port Numbers

Question How can I use a particular TCP/IP port number in Distributed Smalltalk?

Answer The default port number is 1586, which is a registered port number for Smalltalk Distributed. You can use a different port number, if you choose.

1. Change the Activator's port and update the config file; _port=nnnn is the option you need to modify. Or, start up the activator with the -pnnnn parameter
2. Start up the object space (image) with the -dpTCP:nnnncommand line option where "nnnn" is your port number.

See the *VisualAge for Smalltalk Distributed, User's Guide* for more information.

Remedy for a Time-Wasting Method

Tip We found a method that was wasting a lot of time doing nothing. This method is used when you copy an object from one object space to another, like using #asDsByValueObject. The following text can be filed into your image to fix this problem:

Smalltalk

```
DsObjectMobility becomeDefault!
!DsInputStream privateMethods !

reconcileCopy: anArrayOfArraysOfInstances
| beforeList afterList localSpace |

beforeList := OrderedCollection new.
afterList := OrderedCollection new.
localSpace := objectSpace localSpace.
anArrayOfArraysOfInstances do: [ :instArray |
  instArray do: [ :inst |
  inst dsBasicClass reconcile: inst
  afterCopyInto: localSpace
  before: beforeList
  after: afterList ]
].
beforeList isEmpty ifFalse: [
beforeList asArray multiBecome: afterList asArray. ].
```

Using Windows 95 As a Server

Question Can Windows 95 ever be used as a server for Distributed Smalltalk?

Answer Yes, VisualAge for Smalltalk, Distributed technically supports that, but we don't recommend it. Windows has clearly positioned Windows 95 as the client and Windows NT as the server, so the question is how strong a server you would get with Windows 95.

TCP/IP Local Name Server Setup

Tip The Smalltalk Distributed name server requires TCP/IP host names. It resolves TCP/IP addresses considerably faster when host names are entered into the object space descriptor instead of numeric TCP/IP addresses.

As Windows 95 does not have a proper notebook to set up local host names, edit the HOSTS.SAM file in the \WINDOWS directory to contain your local host names and save as file HOSTS (no extension). Entries are one line each; for example:

```
9.127.127.1    LocalHost
9.127.127.2    RemoteHost
9.1.100.200    MyOtherHost
```

Placement of Object Space Security Files

Tip Both object space security files— abtdsusr.cnf for an object space acting as a client or server, and abtdsaut.cnf for an object space acting as a server— must be in installed in the current directory of each of the object spaces. The current directory is usually the directory from which the image is started with abt.exe or ibmst.exe. Both files must be present, even if the default security configuration is used. You can change the placement of the files by setting the security information. See the *VisualAge: Distributed User's Guide* for details.

DBCS Environments

Tip

VisualAge for Smalltalk Version 4 does not support the interoperation of single-byte character set (SBCS) machines with double-byte character set (DBCS) machines in a distributed configuration. Also, Version 4 cannot guarantee that two DBCS machines with different code page mappings will interoperate in a distributed configuration.

Unloading the Distributed Feature

Tip

To unload the distributed feature, follow these steps:

1. Unload all applications that depend on the distributed feature.
2. Reset the Distributed system:
   - Exit the activator daemon program, if it is running.
   - Reset Distributed by selecting **Distribution** → **Options** → **Reset Distributed System.**
3. Close any TrailBlazer browsers, as well as any Distributed tool windows, for example, ObjectExchange and name server browsers.
4. Switch to the standard debugger by selecting **Distribution** → **Options** → **Use Standard Debugger**.
5. Terminate currently running processes. Execute *self halt* to bring up a debugger on the current user interface process. Then use the **Add process** menu item in the debugger to add all processes except AbtEwHoverHelpBackgroundProcess to the debugger. Close the debugger and tell it *not* to keep the processes for future debugging.
6. Ensure that there are no outstanding asynchronous calls, for example, calls made to the operating system from Smalltalk. Then execute the following code from a workspace:

   Smalltalk

   ```
   AcoResourceManger startUp.
   ```

   This step cancels all of the asynchronous callouts. Therefore exercise caution.
7. Execute the following code from a workspace:

```
Smalltalk
DtEventProfilerApplication eventQueueProcessor.
```

8. Unload the feature by selecting **Tools** → **Load** → **Unload Features**. Part way through the unload, the progress indicator window disappears, but you still see status messages in the Transcript.
9. When you get a debugger with the error "undefinedObject doesNotUnderstand close," execute the following code to reset your window system:

```
Smalltalk
'CommonWidgets reinitialize'.
```

10. Check and unload any configuration maps that have not been unloaded in the following order, in case errors occur during feature unload.

From Transcript messages, you should be able to tell whether feature unload is complete and which configuration map has been unloaded:

```
Transcript
Distributed: Example
Distributed: Security
Distributed: VisualAge Part
Distributed: Toolkit
Distributed: Object Space
Distributed: Base Services
AbtObjectMonitorTools
ENVY/Image SCI
```

File Handle Limits on UNIX

Tip

In general, UNIX platforms have file handle limits per operating system process. In VisualAge Distributed, each connection uses two file handles.

Some other operations, such as file operations, also use file handles. Therefore, the number of file handles determines the maximum number of connections in one Smalltalk image. Use *ulimit -n* to check the default number of file handles on your system. Consult your system administrator or user manual to find out how to reset the number of file handles.

Chapter 7. Reports

In this chapter we cover issues related to the VisualAge for Smalltalk Reports feature.

Reports Feature in Version 3 on AIX

Question Why is the Reports Feature not supported on AIX?

Answer Reports didn′t make it onto AIX because the CommonPrinting subsystem hasn′t been implemented yet for the Unix platforms. The subsystem was designed to a proposed desktop printing standard as part of the Common Desktop Environment. The proposal fit the current Motif-based subsystems of CommonGraphics and CommonWidgets, and mapped well to the platform print support on OS/2 and Windows, so it was chosen as the basis for portable print support. Unfortunately, the portable Unix implementation to support this design hasn′t materialized yet.

Given that there would be no print support on AIX in VisualAge for Smalltalk, Version 3, we considered supporting development and viewing of reports on AIX. In the end, we decided that it was better to wait for print support rather than sell the edit/view support as a standalone. At this point, all the possible print solutions from AIX require an OS/2 or Windows machine. This machine could be set up as a print server, and print requests could be made from the AIX clients via the Distributed Smalltalk or another communication protocol. The server would have the reports feature loaded and would have a set of parameterized reports that could be run. For example, it might have a customer detail report that would take a customer object or customer ID as an attribute.

The output of the reports could be printed directly to a server-attached printer, or captured in a Postscript/PCL/other file to be sent back to the AIX client. In the second case, the captured output could be returned to the AIX client to be printed on his locally attached printer. It could also be captured and sent to his printer directly from the server via *lpr* or another remote printing solution. Printing the reports directly on the server would be the easiest way to handle printer configurations. If the reports are going to be printed back on the client machine, then the print server must have to have a print driver installed that is exactly compatible with the printer attached to the client. This might be a problem for some printers. I hope that this design will provide a workable print solution given the available printing platforms.

Win32s Errors with Reports Feature

Tip

If you have experienced Win32s errors when trying to print or preview reports, go to print setup, or print a workspace or transcript, then we have a fix for you. Some print drivers apparently underestimate the space required for their printer-specific driver information, eventually causing memory corruption and a Win32s exception. The fix is to allocate little more space than the drivers ask for. Rather than add a mangled file-in of the long method to be changed, here's a description of the change to be made:

1. Open the application CommonPrintingWin, find the class CgPrintJobAttributes
2. Select class and private methods
3. Find the following method:

   devModeDriver:devName:devMedium:flags:oldDeviceMode:

4. Change the following statement:

```
Smalltalk
devmodeNew := OSDevmode reference:
  (ByteArray new: devmodeSize).
```

   to:

```
Smalltalk
devmodeNew := OSDevmode reference:
  (ByteArray new: devmodeSize+10).
```

The specific printer driver that showed the problem was the IBM 4019 driver, Version 4.3, dated 3/11/94. There could be other problem drivers.

Iterator Field Break

Question

I am using a report iterator field break (AbtReportFieldBreakForm). This class has no extension in AbtRunReportApp, so AbtEditReportApp becomes a prerequisite for my application. This doubles the size of our image, which is a very bad thing. Am I doing something wrong?

Answer To fix it, you have to move the class AbtReportFieldBreakForm from the AbtEditReportApp to the AbtRunReportApp. After this, you have to move the one and only method *visualPolicyClass* back to the AbtEditReportApp (extend the class in the app AbtEditReportApp).

Sums Over Hidden Details

Question Currently we are experimenting with the VisualAge for Smalltalk Report Writer. How can we have sums, and the like over fields not printed in the details line. That is, connecting the *count:* method to an attribute of ″current of iterator″ causes a shift down one line, so that the first value of the new group is counted in the old group and left out of the new one.

Answer You are right about the problem with using a current attribute directly. That′s why we added the *formatted* event from fields. It is only triggered when the field is printed, rather than at iteration which may be signalling a field for the next group. You can connect a script or dependency from *formatted* of a field in the details line to count an un-printed value.

Conditional Printing

Question Is there a way to conditionally omit printing of certain lines? For example, if a group key takes a certain value, we want to omit all detail lines within the group and proceed to the group footer.

Answer For now, there′s no way to do this on a line basis; if a line is present in the body of an iterator it will at least take up some space. You could probably omit lines by adding and removing lines from the body based on the current object′s state. Reports are very dynamic that way—the contents and layout of a line or field or form is not really committed until it hits the page, except for page headers and footers.

General Reports Questions

Question My application has the following reporting requirements:

- A set of canned (standard) reports with headers, totals, breaks, and page controls.
- Reports that can easily be distributed to the requester via Lotus Notes, PROFS (VM mainframe system) or paper, or that persons can easily print/save/manipulate themselves.

- Unscheduled reporting by the programming staff for quick reports or questions.
- Ability to save these unscheduled reports for users, with helps explaining how these reports should be run (quotes, format of parameter input, and so on).
- Structured unscheduled reporting for users, simplistic and quick.
- Ability to choose fields, sorts, and breaks.
- Complex reporting, queries to pull information are structured and simple, however it is not necessarily true that all data will be reported, dependent on multiple program criteria specific to regional information/criteria
- Similar to writing a program with a lot of specs/variables

Does the Reports feature support these requirements? If not, can the parts be easily extended?

Answer Reports are saved as VisualAge for Smalltalk parts. The end-user application would provide access to a set of Reports, and the user would be able to view, preview, or print them. There is no current example of formatting a report to other formats, such as a text file or note, but the design includes the possibility of developing support for other devices.

Although VisualAge for Smalltalk doesn't provide an environment for end-user report editing, one can easily be built into your application. Report layouts are edited in the CompositionEditor and saved as Report parts, but their run-time formatting is very dynamic. Our reports provide mainly layout behavior, with data sources such as Queries or Smalltalk collections responsible for providing data selection, ordering, and the like. End-user report tailoring can be enabled by creating views that tailor the data organization (modify the query's SQL) and modify the layout of the report by adding, removing, or reordering report fields. Attributes such as font and color can also be updated on the fly. In fact, you can create a report completely on the fly if you need to.

To save the unscheduled (ad hoc) reports for users should be straightforward. The report part would be predefined to lay out the results of a query, and the query would be tailored based on the criteria specified.

The reporting capability of the report parts is quite powerful, and is enabled to take advantage of the full capability of our database parts as well as any other Smalltalk objects you want to include in the report. Report parts are

more than a query builder and report generator. The source code is fully available, and they should be easily extendable.

Saving Printer Settings in reportPreview

Question If you access a report for the first time, you have to choose the printer, go into its settings, and define them. Is there an easy way to save default the printer settings?

Answer You could store the printer object in a class variable and connect it to the printer object of the report. Eventually, you could save the printer settings of an AbtReportPrinter using the object dumper. The key attributes are printerDisplay, printerDisplayName, and printerJobProperties. If the file exists, you could restore the attributes from the file, otherwise prompt the user for the printer.

Report Writer Default Fonts

Tip For reasons of National Language Support, it was necessary to change the default font in the Report Writer to a system font. If your existing reports use the default font settings in the Report Writer, you have to take one of the following actions to preserve your existing report layouts:

- Change the default Report Writer font to its Version 3 default:
  1. From the VisualAge Orgainzer window, select **Options** → **Preferences**.
  2. In the VisualAge Preferences window, select the **Report** tab.
  3. Select the **Change default font** button.
  4. In the Font Selection window, change the font to display-Times New Roman-medium-13.
  5. In the Font Selection window, select **OK.**
  6. In the VisualAge Preferences window, select **OK.**

  or
- Set the font in the Report Shell part for each report:
  1. In your report part, open the settings for the Report Shell part.
  2. In the Properties Sheet, select the *deviceFont* field and then select the button that appears.

3. In the Font Selection window, change the font to display-Times New Roman-medium-13.
4. In the Font Selection window, select **OK.**
5. Select **OK** in the Properties Sheet window.
6. Save the report part.

Calculated Fields

Tip

When using calculated fields in a report, you must place the calculated field on a separate line from the field that is being calculated. This avoids timing problems which can cause unpredicatble results in your calculation.

Field Breaks

Tip

When you use field breaks to build a report, ***do not*** delete the watch field for a break. To delete a break, use the **Remove field break...** menu option. Deleting the watch field causes a walkback during testing: ″AbtExternalSubpartBuilder does not understand coElement.″ During run time, it ensures that the break is never triggered.

Packaging

Tip

The Catalog of Classes report includes class and application information which is not available at run time, such as the owner of a class or application, and edition information. Thus it is impossible to package and run this report as a run time application. The IconViewer example from the sample application can be packaged instead.

Be sure to include CLKS.BMP when you package your Report application. Failure to do so will cause a run-time error: ″Primitive failed in: Smalltalk>>#quo: due to Divide by zero in argument 1.″

Using Scripts with Reports

Tip Some sample code has been added to the AbtReportPrinter class to help you select printers from scripts in your reporting application. See the methods *default* and *named:* in the AbtReportPrinter class.

Tip Scripts that are called through attribute-to-script connections can be called before any queries or data initialization. Therefore you code these scripts to handle nil values from uninitialized parts, and you should use the ″More dependencies...″ feature of the attribute-to-script connection′s settings to make sure the field is recalculated when the appropriate data has been updated.

Using Multirow Query Results in Reports

Tip Always check the **Enable packeting** checkbox on the Fetch page of the query′s settings notebook. If this box is not checked, the first row of data might be prefetched and will be skipped when the report is formatted.

Tip Always check the **Read-Only** checkbox on the Update page of the query′s settings notebook. Checking this box improves query performance and prevents possible locking problems.

Tip Override *finalInitialize* in your part to set query host variables for your primary query. Using a connection from *aboutToFormat* might set the host variable after the query has already been run.

Changing the Color of an Iterator Header

Tip If you change the color of an iterator header and then change the color back to the default, the header will appear after the footer. To move the header back to its proper position in the report, drop a new line on the iterator body and then delete the line.

Reporting in an ASCII File for Later Printing

Question How do I write a VisualAge for Smalltalk report in an ASCII file to print it later?

Answer Right now the best way to do this is to print it to a PostScript file using the settings on the print driver to print to a file. You can then print the report later.

Printing in the Report Owner′s Language

Question How do I print reports depending on the language of the report owner?

Answer This should work the same way as views. One problem might be that your report instances might have already been created before you chose the new language. You will have to recreate any reports that might be built with the original language. For example, any embedded reports in the current view would already have been created.

Printing Underlined Words

Question How do I print underlined words with the report writer under OS/2?

Answer We don′t have support yet for underlining, because the base font classes don′t support it.

Counting an Unprinted Value

Question How can we have sums and the like over fields not printed in the details line?

Answer You are right about the problem with using a current attribute directly. That′s why we added the #formatted event from fields. It is triggered only when the field is printed, rather than at iteration which may be signaling a field for the next group. You can connect a script or dependency from #formatted of a field in the details line to count an unprinted value.

Omit Printing of Certain Lines

Question Is there a way to conditionally omit printing of certain lines?

Answer This is a known requirement, and we hope to provide a nice way to do this in the future. For now, there's no way to do this on a line-by-line basis. If a line is present in the body of an iterator, it will at least take up some space. You could probably do this by adding and removing lines from the body based on the current object's state. Reports are very dynamic that way, the contents or layout of a line or field or form is not really committed until it hits the page, except for page headers and footers.

Speeding Report Printing

Question Is there a way to speed up the printing throughput of the report parts?

Answer The only speed-up I can recommend is reviewing the signaling and script execution performed in the report and the objects being reported upon.

Adding Fields to a Report Dynamically

Question How do I add report fields to the report IteratorShell dynamically?

Answer In order to create your fields at the earliest possible moment, you can create them in the method finalInitialize or create them in a script that is invoked when the outer collection is set. When I set the field values, I had a script invoked by the event #self from the current item of the outer collection's iterator. The script filled in the dynamic fields with the items from the inner collection.

HP Printer

Tip Here is a fix that might remedy a problem with your HP printer. The typical symptoms include clipping of tall fonts and jagged right-aligned columns in reports, when using scaled fonts and PCL print streams. The root cause is incorrect results from textWidth. File this in and try it again.

Smalltalk

```
| CgPMPrinterServer privateMethods |
createPS: logAddress driverName: driverName driverData: driverData
"Create a printer OSHdc and an OSHps on it.  Answer the OSHps or
nil."

| devOpenStruct hdc hps pLogAddress pDriverName pDriverData |
pDriverData := OSDrivdata calloc: driverData cb.
driverData memcpyFrom: 0 to: driverData cb - 1 into: pDriverData
  startingAt: 0.
(devOpenStruct := OSDevopenstruc new)
pszLogAddress: (pLogAddress := logAddress copyToOSMemory);
pszDriverName: (pDriverName := driverName copyToOSMemory);
pdriv: pDriverData.

"The DeviceContext for the server must be open OD_INFO to
allow font metrics to be queried without having a print
job current.  OD_QUEUED rejects all gpiCreateLogFont: calls
until a job is actually started."

hdc := Hab
  devOpenDC: OdInfo
  pszToken: '*' "$NON-NLS$"
  lCount: 3
  pdopData: devOpenStruct
  hdcComp: nil.
pLogAddress free.
pDriverName free.
pDriverData free.
hdc isNull
ifTrue: [ ^nil ].
hps := Hab
  gpiCreatePS: hdc
  psizlSize: OSSizel new
  flOptions: PuPels 3 GpifDefault 3 GpiaAssoc 3 GpitNormal.
hps isNull
ifTrue: [
  hdc devCloseDC.
  ^nil ].
^hps
```

Hierarchical Breaks

Question How can I have hierarchical breaks?

Answer All the breaks attached to an iterator are peers, and don't imply any hierarchy. I think it would be pretty easy to add a subclass of CoFieldBreak which had the concept of a parent break, the key method to override is insertBreak, which answers whether or not this break has been triggered. You can probably get results similar to hierarchical breaks by using multiple nested iterators. For example, the hierarchical break report:

Report Breaks

```
CountryStateIterator
CountryHeader: "Country:" countryName!
StateHeader: "State:" stateName!
Body: - state details -
StateFooter: - state totals -
CountryFooter: - country totals -
(break on state key and country key)
```

might produce output like:

Report

```
Country: USA
  - USA details -
State: NC
  - NC details -
  - NC totals -
State: VA
  - VA details -
  - VA totals -
State: FL
  - FL details - < Country break and state break >
  - FL totals -
  - USA totals -
Country: Tonga
  - Tonga detail -
State: Vanuaatu
  - Vanuaatu details - < Country break and state break >
  - Vanuaatu totals -
  - Tonga totals -
```

The same output could be produced from a report with nested iterators for country and state as follows: (Note that the header lines here are actually within the bodies.)

Report Breaks

```
CountriesIterator
CountriesBody
CountryHeader: "Country:" countryName!
StatesIterator  (on states for current Country)
StatesBody
StateHeader: " State:" stateName!
StateDetailsLine: - state details -
StateFooter: - state totals -
CountryFooter: - country totals -
```

It's actually a much cleaner report, which eliminates timing problems with breaks and totals.

Using Break Protocols

Question How does the #addBreak - #removeBreak protocol work?

Answer They take AbtFieldBreak as an argument. Breaks are tested in the order they are added to an iterator. You can add one by creating a new one with a header/footer/watchField/name and passing it to *addBreak:*. You can remove one by name with something like:

Smalltalk

```
iterator removeBreak (iterator subpartNamed: 'badBreak').
```

coElement Role

Question What is the role of the coElement?

Answer CoElements are to AbtReportElements as CwWidgets are to AbtBasicViews. They are the primitive object which knows more about page layout and how to actually render a field. Most report parts correspond to one or more CoElements. See the implementers of coElementClass.

Chapter 8. SOM and DSOM

In this chapter, we cover the SOM implementation for use with VisualAge for Smalltalk. Here are some places to read about SOM and DSOM (in no particular order):

- *Client/Server Programming with OS/2 2.1* (G325-0650-02)

  This tome by Robert Orfali and Dan Harkey should be on the desk of anyone doing client/server programming. It has a great deal of information on SOM Version 1 (as delivered with OS/2 2.0) and some on SOM Version 2 and DSOM(as delivered with SOMobjects for OS/2 and with the VisualAge SOMsupport feature).

- *Client/Server Survival Guide with OS/2* (SR28-5494)

  Orfali and Harkey (above) have followed their original best-seller with another book that is destined for fame. Part 7 has the most relavent information, comprising seven chapters on CORBA, object frameworks (Taligent), OPENDOC vs. OLE/2, DSOM and SOMobjects, and distributed object databases.

- *Object-Oriented Programming Using SOM and DSOM* (SR28-5570)

  Christina Lau provides a soup-to-nuts coverage of SOM and DSOM here. Her programming examples range from introductory to a complete miniapplication. While there's not a lot of depth, some important points are brought out that I have not seen discussed anywhere else.

- *SOMobjects: A Practical Introduction to SOM and DSOM* (GG24-4357)

  This redbook from the IBM ITSO Austin Center takes you through the steps of developing a distributed banking application using DSOM. The book is most appropriate for readers who have some DSOM background.

- *SOMobjects Developers Toolkit Publications* (IBM Publication number is S96F-8649 for the complete set.)

  This set of books contains:

  - User's Guide
  - Programmers Reference
  - Programmers Quick Reference
  - Collection Classes Reference
  - Emitter Framework Guide and Reference
  - Installation/Configuration Guide

- See also, *OS/2 Developer Magazine*. Many past issues have had articles about the SOMobjects implementation of SOM and DSOM. The most recent are:
  - "SOMobjects Developer Toolkit: An Overview," Nov/Dec 93, G362-0001-20
  - "SOMobjects Developer Toolkit: Sharing SOM Objects with DSOM," Jan/Feb 94 G362-0001-21

Generating Classes with a SOM Prefix

Question How do I modify the SOM Smalltalk constructor in V3.0 so that it keeps on generating classes with a SOM prefix?

Answer Changing the way class names are built would be difficult, but not impossible. One thing that makes it hard is that the pattern for naming has to be used globally (since Smalltalk class names are constructed dynamically at run time from the SOM class names). Have you tried simply defining aliases? For example,

```
Smalltalk
Smalltalk at: #SOMMySOMClass put: MySOMClass
```

After doing this, you can refer to the class by either name. You do need to be a bit careful if you take this approach since there are some situations where only the real name will do. For example, the class itself knows only its real name. This means that

```
Smalltalk
SOMMySOMClass name
```

will answer #MySOMClass. If you really want to get the SOMStructure classes as in Version 2, you can still get them in Version 3. You just need to annotate your IDL that defines the structure (in the implementation section) with a statement like this:

```
IDL
ssc_explicit_mapping:  mySOMStruct;
```

Of course, then you are back into the aliasing business again.

SOM Exception

Tip SOM exceptions are transformed to Smalltalk exceptions in Version VisualAge for Smalltalk, Version 3. If you want to handle ***all*** user exceptions, you can code something like the following (the contents of the handler block are up to you, of course):

```
Smalltalk
(SOMGlobals at: 'ExCORBAUser')
  markReadOnly: false;
  defaultHandler:
    [ :signal | Transcript cr; show: 'SOM UserException' ]
  markReadOnly: true.
```

You can use a similar statement (referencing ExCORBASystem) to handle ***all*** system exceptions. If you want special handling for a particular method call, just use the normal Smalltalk exception-handling mechanisms:

```
Smalltalk
[aBlock] when: (AnException) do:
  [:signal | "the exception handling code"].
```

CORBA-Compliant ORBs

Question Which CORBA-compliant ORBs work with a VisualAge for Smalltalk client?

Answer Only IBM SOM works with IBM Smalltalk. There will be wide support for other ORBs in the future as the various ORB-vendors (including IBM) implement CORBA 2.0's ORB Interoperability function.

Using DSOM Hangs the System

Question When I try to access a DSOM object from VisualAge, the system seems to hang. The only way I can get out is to press Ctrl-Esc and end VisualAge. What's happening?

Answer DSOM objects are accessed across a communication link, either interprocess communications (IPC) or a wire. To account for latency in the communications network, the DSOM client must wait for some time before declaring an error due to lack of server response. Since its calls to DSOM are synchronous, VisualAge for Smalltalk waits also. If you have enough patience to wait, DSOM will either return with an exception indication or your VisualAge process will terminate.

If your VisualAge process terminates, the most likely reason is that you didn't have the DSOM daemon running.

You can reduce the wait from its default of 10 minutes to something more reasonable for your working environment by placing a DSOM time-out statement in your CONFIG.SYS file. The value you specify is the number of seconds that DSOM should wait for a response from a server before it declares failure—60 seconds might be a good choice. Here is an example:

```
SET SOMDTIMEOUT=60
```

Adding SOM Objects as VisualAge Parts

Question I am running OS/2 Warp using VisualAge for Smalltalk. Can anyone please tell me what VisualAge software components I need installed and what are the job steps to adding SOM objects that we develop to be used as VisualAge parts? Somehow there is a way to get a menu option in VisualAge that says **Create SOM Wrapper**, but how do I get it?

Answer First you need to have the SOMobjects Developers Toolkit on the machine running VisualAge for Smalltalk Be sure you have compiled the IDL for your classes into the Interface Repository.

Next you need to install the VisualAge SOMsupport. From the Transcript menu **Smalltalk tools**, select **Install** and then select **SOMsupport** in the listbox.

Next you need to create SOM wrapper classes. Open a VisualAge for Smalltalk Applicaion Browser for the application that you want to hold the wrapper classes. From the menu **Classes**, select **Create SOM wrappers**. The dialog panel will show you the available SOM interfaces/modules. Select your interfaces and press **OK**. You need to create the wrapper classes in super/subclass order. This means that all superclasses of the ones you are creating wrappers for must already exist (have wrappers). The wrapper

class names are derived from the SOM interface names by prefixing with SOM (that is, SOMObject in SOM is SOMSOMObject in Smalltalk).

Now you can use the wrapper classes like other VisualAge parts by entering their class name in the Add Part dialog of the Composition Editor. The SOM attributes and operations are available as VisualAge for Smalltalk attributes and actions.

Using DSOM Ends the VisualAge Process

Question When I try to access a DSOM object from VisualAge for Smalltalk, the VisualAge process ends with no indication of the reason. What's happening?

Answer DSOM servers are accessed through an intermediate process known as the DSOM daemon. If this process is not running when VisualAge sends the first DSOM message, SOM raises a terminating error (Error 30109) and ends the invoking process.

You will see the message associated with this error only if you have DSOM debugging enabled (SET SOMDDEBUG=1). The message will be output on STDOUT by SOM (or to the DSOM message file if it is enabled). You must have STDOUT redirected to a file to see its content.

You can ensure that the DSOM daemon is running by including it in your STARTUP.CMD file. Enter a line similar to the following:

```
start "DSOM Daemon" /MIN somdd.exe
```

Environment Variable Setup for SOM

Question Can you summarize for me the setup of environment variables in my CONFIG.SYS file for using SOM with VisualAge for Smalltalk?

Answer You can find detailed descriptions about the environment variables setup in the *SOMobjects Base Toolkit Users Guide* which is shipped with the VisualAge for Smalltalk SOMsupport component. The summary below shows how we have set up our environment variables for successful operation. If you use any of the suggested environment variable settings, you should replace 'C:\SOM' with the drive and directory that your

SOMBASE environment variable is set to and 'C:\VISUALAG' with the base VisualAge drive and directory.

- SOMIR— Specifies the list of interface repositories to be used. The default statement generated in your CONFIG.SYS file during installation of the SOMobjects product specifies two .IR files:

  ```
  SET SOMIR=C:\SOM\ETC\SOM.IR;SOM.IR
  ```

  You will almost always want to change this statement. For reliable operation of SOM, the statement should contain only fully qualified library names.

  If you want to group all your .IR files in one place, you might change it to something like the following:

  ```
  SET SOMIR=C:\SOM\ETC\SOM.IR;C:\SOM\ETC\MYSOM.IR
  ```

  If you want to group your personal extensions to the system .IR together with the rest of the VisualAge for Smalltalk, you might change it to something like the following:

  ```
  SET SOMIR=C:\SOM\ETC\SOM.IR;C:\VISUALAG\MYSOM.IR
  ```

  See the *SOMobjects Base Toolkit Users Guide*, 4-46, 6-9, 6-40, and 7-3

- SOMDDIR— Specifies the directory where various DSOM files are stored (in particular, the implementation repository). Although SOM uses a default value if you don't specify this environment variable, we have experienced some problems that seemed to disappear when we set the variable explicitly:

  ```
  SET SOMDDIR=C:\SOM\ETC\DSOM
  ```

  See the *SOMobjects Base Toolkit Users Guide*, 6-9 and 6-40

- SOMDTIMEOUT— Specifies how long a DSOM client should wait for an acknowledgment. If you are using local (cross-process) DSOM, you might specify a very short wait time. If you are communicating with DSOM servers across a network, you probably want a longer wait time. In any case, the default wait time of 10 minutes is probably way too long for your environment. We use 60 seconds as our time-out value:

  ```
  SET SOMDTIMEOUT=60
  ```

  See *SOMobjects Base Toolkit Users Guide*, 6-41

- SOMDDEBUG— May be set to enable DSOM run-time error messages. If it is not set, or if it is set to zero, error messages are not issued and the only indication of an error is the exception code returned from a method invocation. If you do enable run-time error messages, you may receive some false messages (sometimes an error is an expected result of an operation), so you should carefully sort the false errors from the real

ones. You might want to run with messages enabled while developing applications using SOM and with messages disabled in your deployed applications:

```
SET SOMDDEBUG=1
```

See *SOMobjects Base Toolkit Users Guide*, 6-41 and 6-68

- SOMDMESSAGELOG— Specifies the name of a file where DSOM run-time error messages are recorded. If it is not set, the messages are written to the standard output device. Setting this variable is most useful if you want to capture debugging messages (see SOMDDEBUG above) in a deployed application rather than showing them to the application's users. Note that, due to a bug in SOM, you cannot include a drive when specifying the value for this variable:

  ```
  SET SOMDMESSAGELOG=\VISUALAG\SOMDMSG.LOG
  ```

  See *SOMobjects Base Toolkit Users Guide*, 6-41 and 6-68

Error: somFindClass failed for class Xxxxx

Question When I try to run an application using SOM objects, I get a Transcript error message 'somdFindClass failed for class Xxxxx.' What is wrong?

Answer The usual cause of this error is one of the following:

1. The class definition could not be found in any of the accessed Interface Repositories.

   The accessed Interface Repositories are specified in the SOMIR environment variable. If you think you have things set up properly, try using the IRDUMP utility to look at the class. From the same drive and directory where you start VisualAge, use this.

   ```
   IRDUMP classname
   ```

   If the interface definition for the class is not displayed, you have an error in your set up.

2. The DLL containing the implementation of the class could not be found in any of the directories specified in your LIBPATH statement.

   a. If the DLL is not in a directory specified in your LIBPATH, either update your LIBPATH and reboot, or move the DLL file.

   b. The mapping from the interface definition to a DLL may be incomplete or incorrect. If the name of your SOM class is not the same as the name of the DLL that implements it, you normally must

specify the dllname= modifier in the implementation section of the IDL file for the class (see *SOMobjects Base Toolkit Users Guide*, 4-32). The only exception to this requirement occurs when you manage your own class-to-implementation-file mapping using the SOMClassMgr>>#somFindClsInFile techniques.

SOM Support Feature

Question Is there a way to set up and use the VisualAge for Smalltalk SOM feature without the SOM Development Toolkit?

Answer The VisualAge for Smalltalk SOM support provides the needed subset of the SOM objects Developer's Toolkit. It provides the function you need to add SOMobjects to your interface repository. Install only the SOMobjects Base Toolkit that is shipped with VisualAge for Smalltalk.

SOM Methods with Inout Sequences

Question I have some methods in my SOM classes that have inout sequences as parameters. Smalltalk maps sequences to OrderedCollections with the _maximum field of the sequence set to zero. But according to SOM documentation, the _maximum member of the sequence in an inout parameter must be set to the actual size of the buffer. This seems reasonable so that the SOM object knows how much memory is available for the answer.

Will there be any way of setting the maximum field to something different than zero (for instance, the actual size of the OrderedCollection)?

Answer It actually works just a bit different than is described, but the effect is the same (unbounded inout sequences are sent with a maximum of zero). In Smalltalk, maximum is a computed value (it is not stored with the OrderedCollection). Therefore, it can have one of two values: the maximum specified in the IDL (bounded), or the current size of the OrderedCollection (unbounded). Of course, the unbounded case is not working.

The patch is fairly simple. You need to update SOMTC>>#toSOMMemory:at: by finding the code block that handles sequences (search for TCSequence). Update the code block by changing:

```
Smalltalk
maximum = 0 ifTrue: [length := length min: maximum].
```

to:

```
Smalltalk
maximum = 0
  ifTrue: [length := length min: maximum] "bounded sequence"
  ifFalse: [maximum := length].  "unbounded sequence"
```

SOM Objects

Question Is it possible to create VisualAge for Smalltalk parts out of SOM objects?

Answer Yes, you can use SOM objects as VisualAge for Smalltalk parts. First you need to generate the SOM wrappers; then you can use the Add Part interface in the Composition Editor to add the SOM object to your application. Look at the SOMDSamplesApp in VisualAge for Smalltalk for some examples.

Wrapper generation is a two-step process. In the first step, you use the SOM Compiler to create entries for your interfaces in the SOM Interface Repository (IR). In the second step, you generate the SOM wrappers (language bindings) in Smalltalk. You can do this either by using the wrapper-generation programming interface or by using the Generate SOM Wrappers user interface accessed through the VisualAge for Smalltalk Organizer. This second step reads the interface descriptions from the SOM IR built in the first step. VisualAge for Smalltalk SOM support also uses the SOM IR at run time to retrieve data-marshaling information.

ABT.SOM.1017.e: #somFindClass Failed

Question I received the following message: ABT.SOM.1017.e: #somFindClass failed; 'dep' could not be loaded or class 'CDep' could not be initialized. If I inspect the components of the part I am testing, I find the the window and nil (instead of the initialized SOM wrapper). Any help?

Answer There are two likely causes for this problem:

1. SOM could not find the DLL containing CDep, or a DLL containing a class that CDep subclasses, by searching the subdirectories listed in LIBPATH. One way to check if this is your problem is to get the CHK4DLLS package from OS2TOOLS or run CHK4DLLS from NETDOOR. From the subdirectory that you are running VisualAge from (this is important), run:

   ```
   chk4dlls -l dep.dll
   ```

   If CHK4DLLS doesn't complain, this probably is not your problem.

2. SOM could not create a class instance for CDep (or a class that CDep subclasses). Look at the link map for all your SOM DLLs (dep.dll and any others that you wrote and which contain SOM classes). Each of the DLLs should export SOMInitModule (spelling counts!). Look in Chapter 5.6 of the *SOMobjects User's Guide* for information on writing a SOMInitModule function.

Note also that it is normal to get a walkback for "UndefinedObject doesn't understand ....." associated with this message. When the class cannot be created, a nil object is answered by the SOM method _somFindClass. Eventually someone tries to do real work with the nil object and the walkback results.

SOM Objects on OS/2 Desktop

Question How can I send messages to SOM objects that live on an OS/2-desktop? For example, I'd like to send wpHide to the launchpad. Is there someone who has something like this running? I tried it, but it doesn't work. When I run the part I made, I get the following message in the transcript (despite copying the pmwp.dll in my VisualAge for Smalltalk directory).

> ABT.SOM.1017.e: #somFindClass failed; 'pmwp' could not be loaded or class 'WPLaunchPad' was not initialized.

Here is what I did:

- I loaded the SOMSupport feature into my image
- I generated SOM wrappers for WPObject, WPAbstract, WPLaunchPad
- I created a new visual part, added WPLaunchPad as a part, and connected the wpHide action with a button-clicked event.

Answer To work with the OS/2 Workplace Shell objects, you need to have the *OS/2 Warp Workplace Shell Programming Guide* and the *OS/2 Warp Workplace Shell Programming Reference* at hand.

OS/2 Workplace Shell Objects are special. They reside in their own framework under WPObject and don't respond as you might expect to the normal SOM messages. For example, you can't create an instance of one with #somNew; you must use #wpclsNew. But you don't really want a new instance of the LaunchPad, you want the one that's on the desktop already. So, I think you need a script that will send #wpclsFindObjectFirst with the correct parameters to WPObject.

Chapter 9. Web Connection

In this chapter, we present a set of questions and answers on the Web connection feature.

Retaining State between Requests

Question Does the VisualAge for Smalltalk Web Connection feature have any features that make it possible to maintain server state between related requests? The classic example of this, I guess, would be the *web shopping basket* into which users pop different items until they are ready to commit the order.

Answer You should use something called Session Data. Session data is basically a value holder with an expiration time-out. For VisualAge for Smalltalk, you specify the class name of the part, and then we'll give you that part as a tear-off for you to do with what you want. Expiration time-outs are configurable. A session key is generated for every session data object, which is sent between the browser and VisualAge for Smalltalk via hidden input fields.

Disabling a Button

Question I have two buttons on an HTML form. The first button must be enabled while the second button must be disabled. When clicking on the first button I want to enable the second and disable the first. Is it possible to do that with Web parts?

Answer There is no way to disable anything in HTML. About the only thing you can do is to destroy the button programmatically. Unfortunately, *destroyPart* won't cut it. Instead, you should (using a script), send the message *removeFromParentPart* to the part you want to remove from the generated HTML.

GIFs not Displayed

Question I don't seem to be able to get images displayed. In my browser. The image outline is briefly displayed, but no image appears. I put LOGO.GIF in the same directory as my home page file, INDEX.HTM, I also put it in the VisualAge for Smalltalk directory. If I reference it in the home page, it is displayed; similarly, if I save the generated HTML and load the file into the browser it is displayed.

A cut-down example page generates the following:

HTML

```
<!-- generated by VisualAge Web Connection on 18-06-96 -->
<html><head><title>Test Page</title></head>
<body>

<h1><font color="#FF0000">Hello</font></h1>
<img src="logo.gif" alt="">

</body>
</html>
```

I assume there is something incorrect in my configuration. Any ideas?

Answer From the HTML, it looks like you are specifying the run-time image name as LOGO.GIF. Now, let's assume that I got to your page by way of the universal resource locator (URL) http://server/cgi-bin/abtcgil.exe/MyPart. LOGO.GIF is a relative URL which will be looked for as http://server/cgi-bin/abtcgil.exe/logo.gif. Wrong! What you probably want is to specify the URL as /LOGO.GIF, which will be interpreted as http://server/logo.gif.

Handling Pseudo-Pages

Question We need to provide the user with a dynamic list of pages, like the PartLister, where in fact each page will be the same VisualAge for Smalltalk page, but with dynamically created different contents.

Viewing any page causes the links to ***all*** the pages to change color.

This is handled by Alta Vista Searcher by suffixing parameters to the URL, so that the URLs all look different to the browser. Are these parameters, the

string of name=value pairs after the ? in the URL, related to the HTML Form Data? Are there any supplied methods for parsing this parameter string?

Answer You can access form data as you suggest via the formData attribute of the request part. Probably the easiest thing to do is to drop a composite part next to the page part with the links on it, drop a form in that, and drop an entry field for every parameter you want to access. These entry fields should be named the same as the parameter you are going to pass in. Save the part. Now, you can access the parameters by using a form data part pointing to the part with the page and composite combo on it. The composite exists merely so that you can add attributes to the form data for this page. It will never be used.

For example, let's say you want to create two text links on a page pointing to some other part, but pass different parameters to them. Call the parts Part1 and Part2. Call the parameters parm1 and parm2. On the text parts on the page part of Part1, specify URL for the link, and type in

```
Part2?parm1=yes&parm2=no
   and
Part2?parm2=no&parm1=yes
```

for the links of the two text parts. In the composite, you should have a form with two entry fields named parm1 and parm2. Now, on Part2, drop a form data part, specify Part1 as the part to get the data from. You can now access the *yes* and *no* values via the parm1 and parm2 attributes of the form data.

Packaging Web Application

Question Which files do I need to include with my packaged web application?

Answer The run-time files needed for Web Connection, in addition to the standard files required by VisualAge for Smalltalk are as follows:

- MPR files
  - abtwce30.mpr
  - abtwve30.mpr
  - abtwre30.mpr
  - abtcpe30.mpr (TCP/IP)
- CAT files
  - cfs.cat

  - esta.cat
  - esw.cat
  - cp.cat
  - krn.cat
  - esd.cat
  - cw_e.cat
  - em.cat
  - cpswin.cat
- Other TCP/IP files
- abttcp30.dll (OS/2 and Windows)
- abttcp30.w (AIX)

HTML Links and Session Data

Question I have a Web page with links to other HTML parts. When I click on a link, the second page should be opened with a dynamically generated HTML list. To generate the HTML, I try to pass session data to the second page.

The problem is that I am only using literal HTML text, so I cannot check the Use session data check box. The result is that the CGI session data is always empty. How do I get around this?

Answer You will need to pass the current session key as part of the URL you are building in the dynamic list. You can do this by appending the part name in the URL with a *"-"* (hyphen) followed by the current session data key, with no intervening spaces. The session data key for a given part can be retrieved from the current transaction by using the message *self transaction sessionDataKey*, for example,

Smalltalk

```
partNameWithKey := 'MyPartName', '-',
   self transaction sessionDataKey.
```

URL Query String

Question Can I suppress the query string that is passed back to the browser after a request has been made of the server? I've got a lot of stuff being passed around in the URL and it is not very visually pleasing to have things like randomly generated session keys and form data being visible to the user in the URL field of their browser.

Answer On the settings page for the HTML Form part there is a setting for the method of submission. If you select Get, the form data is passed back in the URL and if you select Post, the form data is passed back via hidden input fields. Sounds like you might have selected the Get submission option; if so, selecting Post should give you what you want.

On the other hand, if you are not using forms, but using image and text links instead, there is no way to suppress the URL shown to the user, other than by turning off the URL entry field (most browsers allow you to turn it off).

Session Data Lifetime

Question I am using WebConnection in a plain Smalltalk environment. Is there a way to change the lifetime of session data to a new value, each time one is created?

Answer You can change the default (15 minutes) via the class message

```
Smalltalk
AbtCgiLinkSessionData class>>defaultLifeTimeSeconds: anInteger
```

To change the lifetime of a session data object that already exists, use

```
Smalltalk
sessionData
    lifeTimeSeconds: secondsThatItWillLiveFor;
    touch
```

Parts Usable with the Web

Question What VisualAge for Smalltalk parts are usable with the Web connection?

Answer All the nonvisual parts currently available in VisualAge for Smalltalk are usable from within VisualAge for Smalltalk Web Connection.

GUI Differences

Question Why do the GUIs built with the Web parts appear differently in different browsers?

Answer HTML is provided to the browser. How the browser displays this tagged HTML text depends on how IBM, or NetScape, or Microsoft built the browser program. The charm of a browser is that it is an application can be deployed across many platforms. The challenge of a browser is that how the application looks to the end user cannot be guaranteed.

Packaging AbtWebSamplesApp and AbtChatSampleApp Separately

Tip When packaging the Web Connection samples, package AbtWebSamplesApp and AbtWebChatSampleApp separately. Although AbtWebChatSampleApp is apparently packaged with the AbtWebSamplesApp, an error in the packaging rules prevents some chat classes from being included in the general samples package. If you package the chat sample separately, it will function properly.

One way to fix this is to use the Browse Packaged Image action from the Transcript tools pulldown and package the sample with this interface. In this case, in the Applications and ICs page, select both AbtWebSamplesApp and AbtWebChatSampleApp. Then use **Reduce** and **Output Image** normally.

Unloading Web Connection Feature after Running WSI Servers

Tip Here is the procedure for unloading the Web Connection feature after running WSI servers.

1. Stop any running WSI servers, using the Web Server Interface Monitor.
2. Save and exit the image. Restart the image. This action cleans up the TCP objects in the Smalltalk image.

3. Unload the feature from the Load/Unload features interface.

Using DBCS Fonts on Windows Platforms

Tip

In DBCS environments on Windows platforms, the default font used after installation may cause a walkback when you try to edit it from the Web Connection page in the Preferences notebook.

To bypass the walkback, execute the following code:

Smalltalk

```
AbtHtmlFontManager class classPool
at: 'ProportionalFontName' put: CgFontStruct default name.

AbtHtmlFontManager class classPool
at: 'MonospacedFontName' put: CgFontStruct default name.
```

After executing this code, you should be able to edit the fonts from the Web Connection page in the Preferences notebook.

Double-Byte Part Names

Tip

Some browsers cannot parse HTML that references DBCS names. Therefore, for Web Connection, do not use DBCS characters in your part names.

Appendix A. Special Notices

This publication is intended to help VisualAge for Smalltalk developers avoid common programming pitfalls and as a guide for frequently asked questions about the IBM Smalltalk language, VisualAge for Smalltalk, and its features. The information in this publication is not intended as the specification of any programming interfaces that are provided by VisualAge for Smalltalk. See the PUBLICATIONS section of the IBM Programming Announcement for VisualAge for Smalltalk for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and

integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AIX/6000
Application System/400	APPN
AS/400	CICS
Common User Access	CUA
DATABASE 2	DB2
DB2/2	DB2/400
DB2/6000	IBM
IMS	MQ
MQSeries	MVS
MVS/ESA	Object Connection
OS/2	OS/400
Presentation Manager	PS/2
RPG/400	SOM
SQL/400	TalkLink
VisualAge	VisualGen
VisualInfo	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.
PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.
UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.
Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

HP, HP-UX	Hewlett-Packard Company
Sun, Solaris	Sun Microsystems, Inc.
Lotus, Notes	Lotus Development Corporation

Other trademarks are trademarks of their respective companies.

Appendix B. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 263.

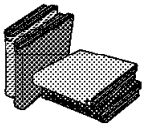
- *VisualAge: Concepts and Features*, GG24-3946
- *VisualAge and Transaction Processing in a Client/Server Environment*, GG24-4487
- *AS/400 Application Development with VisualAge for Smalltalk*, SG24-2535
- *World Wide Web Server Development with VisualAge for Smalltalk*, SG24-4734
- *VisualAge: Building GUIs for Existing Applications*, GG24-4244
- *VisualAge for Smalltalk Distributed*, SG24-4521
- *VisualAge for Smalltalk and SOMobjects*, SG24-4390
- *OO Programming with Client Access for OS/400 and ODBC using VisualAge for Smalltalk*, SG24-4718
- *Application Development with VisualAge for Smalltalk and MQSeries*, SG24-2117
- *Object-Oriented Application Development with VisualAge for C++ for OS/2*, SG24-2593
- *Programming with VisualAge for C++ for Windows*, SG24-4782
- *IBM VisualAge for Cobol for OS/2: Workframe User Guide*, SG24-4604
- *IBM VisualAge for Cobol for OS/2: Primer*, SG24-4605
- *IBM VisualAge for Cobol for OS/2: Object-Oriented Programming*, SG24-4606
- *Visual Modeling Technique, Object Technology using Visual Programming*, SG24-4227

Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

Other Publications



These publications are also relevant as further information sources:

- *VisualAge for Smalltalk User's Guide*, SC34-4518
- *VisualAge for Smalltalk Programmer's Guide to Building Parts for Fun and Profit*, SC34-4496
- *IBM Smalltalk Programmer's Reference*, SC34-4493
- *IBM Smalltalk User's Guide*, SC34-4536
- *IBM Smalltalk: The Language*, by David N. Smith, Benjamin/Cummings Publishing Company, ISBN: 0-8053-0908
- *VisualAge and Transaction Processing in a Client/Server Environment*, by Andreas Bitterer, Michel Brassard, William Nadal, and Chris Wong, Prentice Hall PTR, 1996, ISBN: 0-13-460874-7
- *AS/400 Application Development with VisualAge for Smalltalk*, by Andreas Bitterer, Masahiko Hamada, John Oosthuizen, Gino Porciello, and Håkon Rambek, Prentice Hall PTR, 1997, ISBN: 0-13-520453-4
- *World Wide Web Programming: VisualAge for C++ and Smalltalk*, by Andreas Bitterer and Marc Carrel-Billiard, Prentice Hall PTR, 1997, ISBN: 0-13-612466-6
- *Object-Oriented Application Development with VisualAge for C++ for OS/2*, by Marc Carrel-Billiard, Peter Jakab, Isabelle Mauny, and Rainer Vetter. Prentice Hall PTR, 1996, ISBN: 0-13-242447-9
- *Programming with VisualAge for C++ for Windows*, by Marc Carrel-Billiard, Michael Friess, and Isabelle Mauny, Prentice Hall PTR, 1997, ISBN: 0-13-618208-9

- *VisualAge for Smalltalk Distributed*, by Walter Fang, Sven Guyet, Randy Haven, Matti Vilmi, and Eduardo Eckmann, Prentice Hall PTR, 1996, ISBN: 0-13-570805-2
- *VisualAge for Smalltalk SOMsupport*, by Walter Fang, Raymond Chu, and Markus Weyerhäuser, Prentice Hall PTR, 1997, ISBN: 0-13-570813-3
- *IBM Smalltalk Programming for Windows and OS/2*, by Dan Shafer and Scott Herndon, Prima Publishing, 1995, ISBN: 1-55958-749-0
- *Visual Modeling Technique*, by Daniel Tkach, Walter Fang, and Andrew So, Addison-Wesley, 1996, ISBN: 0-8053-2574-3
- *Smalltalk with Style*, by Edward Klimas, Suzanne Skublics, and David Thomas, Prentice Hall PTR, 1996, ISBN: 0-13-165549-3
- *Object Technology in Application Development*, by Daniel Tkach and Richard Puttick, Benjamin/Cummings Publishing Company, ISBN: 0-8053-2572-7
- *Designing Object-Oriented Software*, by Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener, Prentice Hall PTR, 1994, ISBN: 0-13-629825-7
- *TCP/IP Tutorial and Technical Overview*, by Eamon Murphy, Steve Hayes, and Matthias Enders, Prentice Hall PTR, 1995, ISBN: 0-13-460858-5
- *Object-Oriented Interface Design: IBM Common User Access Guidelines*, SC34-4399

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

A

abstract class. A class that provides common behavior across a set of subclasses but is not itself designed to have instances that work.

abstraction. A simplified description or view of something that emphasizes characteristics or purposes relevant to the user while suppressing details that are immaterial or distracting.

accessor methods. Methods that an object provides to define the interface to its instance variables. The accessor method to return the value of an instance variable is often called a *get* method or *getter* method, and the accessor method to assign a value to an instance variable is called a *set* method or *setter* method.

access plan. The control structure produced during compile time that is used to process SQL statements encountered when the program is run.

action. In VisualAge, a function or operation that a part can perform upon receiving a message. Actions enable a part′s public interface to give other parts access to its behaviors. Compare to *event* and *attribute*.

activate. To make a resource of a node ready to perform the functions for which it was designed. Contrast with *deactivate*.

activator. In distributed processing environments, a program running in the background (sometimes called a ″daemon process″) that initially activates other programs the first time they are called.

active. (1) Able to communicate on the network. A token-ring network adapter is active if it is able to transmit and receive on the network. (2) Operational. (3) Pertaining to a node or device that is connected or is available for connection to another node or device. (4) Currently transmitting or receiving.

adapter. Hardware card that allows a device, such as a PC, to communicate with another device, such as a monitor, a printer, or other I/O device.

address. (1) In data communication, the IEEE-assigned unique code or the unique locally administered code assigned to each device or workstation connected to a network. (2) A character, group of characters, or a value that identifies a register, a particular part of storage, a data source, or a data sink. The value is represented by one or more characters. (3) To refer to a device or an item of data by its address. (4) The location in the storage of a computer where data is stored. (5) In word processing, the location, identified by the address code, of a specific section of the recording medium or storage.

Advanced Program-to-Program Communication (APPC). (1) IBM′s architected solution for program-to-program communication, distributed transaction processing, and remote database access. A transaction program (TP) using the APPC API can communicate with other TPs on systems that support APPC. (2) An implementation of the Systems Network Architecture (SNA) logical unit (LU) 6.2 protocol that enables interconnected systems to communicate and share the processing of programs.

agent. A VisualAge part used to encapsulate business logic and data access executed outside a VisualAge for Smalltalk image.

allocate. A logical unit (LU) 6.2 application program interface (API) verb used to assign a session to a conversation for the conversation′s use. Contrast with *deallocate*.

American National Standard Code for Information Interchange (ASCII). The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among

data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphics characters.

analog. Pertaining to data consisting of continuously variable physical quantities. Contrast with *digital*.

anonymous FTP. Using the FTP function of the Internet anonymously by not logging in with an actual, secret login ID and password. Often permitted by large, host computers that are willing to share openly some of the files on their system to outside users who otherwise would not be able to log in. See also FTP.

API. See *application program interface*.

APPC. See *Advanced Program-to-Program Communication*.

application. (1) The use to which an information processing system is put; for example, a payroll application or an order-entry application. (2) A collection of defined and extended classes that provides a reusable piece of functionality. An application contains and organizes functionally related classes. It also can contain subapplications and specify prerequisites.

application layer. In Open Systems Architecture, the layer of the OSI reference model that provides a means for application processes residing in different open systems to exchange information.

application manager. (1) A team member who is responsible for the overall state of an application. An application manager coordinates the activities of the application's developers and assigns ownership of classes to team members. (2) The browser from which users can create, delete, manage, or configure applications in their image.

application program. (1) A program written for or by a user that applies to the user's work. Some application programs receive support and services from a special kind of application program called a network application program. (2) A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

Note: Do not use the term *application* in place of *application program.*

application program interface (API). An architected functional interface supplied by an operating system or other software system. The interface enables an application program written in a high-level language to use specific data or functions of the underlying system.

application programmer. A person who primarily writes and modifies programs for application data. Contrast with *system programmer*.

Archie. An Internet tool for finding files stored on *anonymous FTP* sites. You need to know the exact file name or a substring of it.

architecture. A logical structure that encompasses operating principles including services, functions, and protocols.

argument. A data element included as part of a message. Arguments provide additional information that the receiver can use to perform the requested operation. Binary messages and keyword messages take arguments. In a keyword message, a colon (:) following a keyword indicates that an argument is required.

ARPANet. (Advanced Research Projects Agency Network), the precursor to the *Internet*. Developed in the late 1960s and early 1970s by the US Department of Defense as an experiment in wide-area-networking that would survive a nuclear war. See also Internet.

array literal. A literal that is an indexed sequence of literals. The symbol # precedes this sequence and parentheses enclose the sequence. For example, #(5 7 9) is an array of three integers.

ASCII. (American Standard Code for Information Interchange), this is the world-wide standard for the code numbers used by

computers to represent all the upper and lower-case Latin letters, numbers, punctuation, etc. There are 128 standard ASCII codes each of which can be represented by a 7-digit binary number, 0000000 through 1111111.

atomic. (1) Pertaining to the smallest element in a composite object that can be manipulated independently. (2) Pertaining to a set of operations performed such that either all the operations performed or none of the operations are performed. (3) Pertaining to a set of operations that cannot be interrupted, such as a critical section of a parallel program.

attribute. (1) In VisualAge, data that represents a property of a part. (For example, a customer part could have a name attribute and an address attribute.) Attributes enable a part's public interface to give other parts access to its properties. An attribute can itself be a part, with its own behavior and attributes. Compare to *event* and *action*. (2) Information that describes the characteristics of system objects or program objects.

attribute-to-attribute connection. A connection from an attribute of one part to an attribute of another part. When one attribute is updated, the other attribute is updated automatically. See also *connection*.

attribute-to-script connection. A connection from an attribute of a part to a script. The connected attribute receives its value from the script, which can make calculations based on the values of other parts. See also *connection*.

authority. The right to do something on the system or to use an object, such as a file or document, in the system.

authorization list. A list that gives a group of users one or more types of access to objects (such as files or programs) or data in the objects (such as records in a file). It consists of a list of two or more user IDs and their authorities for system resources.

B

backbone. A high-speed line or series of connections that forms a major pathway within a network. The term is relative, as a backbone in a small network is likely to be much smaller than many nonbackbone lines in a large network. See also Network.

bandwidth. The transmission capacity of the lines that carry the Internet's electronic traffic. Historically, it's imposed severe limitations on the ability of the Internet to deliver all that we are demanding it deliver, but fiber-optic cables will ensure that bandwidth soon will be essentially limitless and free.

Basic Input/Output System (BIOS). In IBM personal computers with PC I/O channel architecture, microcode that controls basic hardware operations such as interactions with diskette drives, fixed disk drives, and the keyboard.

baud. In common usage the baud rate of a modem is how many bits it can send or receive per second. Technically, baud is the number of times per second that the carrier signal shifts value; for example a 1200 bit/second modem actually runs at 300 baud, but it moves 4 bits per baud. See also bit, modem.

behavior. (1) The set of external characteristics that an object exhibits. (2) The abstract class that provides common behavior for class and metaclass objects.

binary. (1) Pertaining to a system of numbers to the base two; the binary digits are 0 and 1. (2) Pertaining to a selection, choice, or condition that has two possible different values or states.

bind. The process by which the output from the SQL precompiler is converted to a usable structure called an *access plan*. This process is the one during which access paths to the data are selected and some authorization checking is performed.

bit. (binary digit) A single digit number in base 2, in other words, either a 1 or a zero. The smallest unit of computerized data. *Bandwidth* is usually measured in bits per second. See also bandwidth, bps, byte, kilobyte, megabyte.

block. A Smalltalk object consisting of one or more statements, enclosed in brackets [ ], passed as arguments or used as the receiver of messages that implement control flow. Blocks can define temporary variables for their own use.

Bps. (bits per second) A measurement of how fast data is moved from one place to another. A 28.8 modem can move 28,800 bits per second. See also bandwidth, bit.

browse. A way of looking at a file that does not allow you to change it.

browser. (1) A window that supports one or more programming activities, such as creating new classes or methods, modifying existing classes or methods, or viewing library members. (2) Software that enables users to browse through the cyberspace of the World Wide Web. See also Client, URL, WWW.

buffer. (1) A portion of storage used to hold input or output data temporarily. (2) A routine or storage used to compensate for a difference in data rate or time of occurrence of events, when transferring data from one device to another.

byte. A set of bits that represent a single character. Usually there are 8 bits in a byte, sometimes more, depending on how the measurement is being made.

C

cascaded messages. Multiple messages sent to the same receiver object. The messages are separated by a semicolon (;).

category. (1) On the VisualAge Composition Editor, a selectable grouping of parts represented by an icon in the left-most column. Selecting a category displays the parts belonging to that category in the next column over. (2) A logical association of a group of methods within a class, with a name assigned by the class developer.

CGI Link. A stand-alone executable program that receives incoming CGI requests and routes them to the VisualAge application. CGI Link runs on the HTTP server, which does not have to be the same as the machine running the VisualAge application.

CGI Link session data. A Web Connection nonvisual part that holds a persistent data object. You can use CGI Link Session Data to store an application-specific object that remains valid from one CGI query to the next, for the duration of a **session**.

change-event symbol. In VisualAge, the code used to signal that an attribute has changed in value.

character. A symbol used in printing. For example, a letter of the alphabet, a numeral, punctuation, or any other symbol that represents information.

character literal. A literal that is any single character preceded by a dollar sign ($).

CICS. See *Customer Information Control System*.

class. The specification of an object, including its attributes and behavior. Once defined, a class can be used as a template for the creation of object instances. *Class,* therefore, can also refer to the collection of objects that share those specifications. A class exists within a hierarchy of classes in which it inherits attributes and behavior from its superclasses, which exist closer to the root of the hierarchy. See also *inheritance, metaclass, polymorphism, private class* and *public class*.

class definition. The definition of a class, containing:

- Class name

- Type of class
- Immediate superclass for the class
- Instance, class, and class instance variables
- Pool dictionaries that the class uses.

class developer. A team member who develops and changes classes. The team member who created an edition of a class is that edition′s class developer. Contrast with class owner.

class extension. An extension to the functionality of a class defined by another application. The extension consists of one or more methods that define the added functionality or behavior. These methods cannot modify the existing behavior of the defined class; they can only add behavior specific to the application that contains the extended class.

class hierarchy. A tree structure that defines the relationships among classes. A class has subclasses down the hierarchy from itself and superclasses up the hierarchy from itself. The methods and variables of a class are inherited by its subclasses.

class instance variable. Private data that belongs to a class. The defining class and each subclass maintain their own copy of the data. Only the class methods of the class can directly reference the data. Changing the data in one class does not change it for the other classes in the hierarchy. Contrast with *class variable*.

class method. A method that provides behavior for a class. Class methods are usually used to define ways to create instances of the class. Contrast with *instance method*.

class owner. Team member responsible for the integrity of that class in an application edition. The class owner is responsible for releasing class versions. Contrast with class developer.

class variable. Data that is shared by the defining class and its subclasses. The instance methods and class methods of the defining class and its subclasses can directly reference this data. Changing the data in one class changes it for all of the other classes. Contrast with *class instance variable*.

client. A software program that is used to contact and obtain data from a server software program on another computer, often across a great distance. Each client program is designed to work with one or more specific kinds of server programs, and each server requires a specific kind of client. A Web browser is a specific kind of client. See also browser, server.

client object. An object that requests services from other objects.

client/server. The model of interaction in distributed data processing in which a program at one location sends a request to a program at another location and awaits a response. The requesting program is called a client, and the answering program is called a server.

code page. A font component that associates code points and character identifiers. A code page also identifies how undefined code points are handled.

collection. (1) In Smalltalk, a set of elements in which each element is an object. (2) In SQL, a set of objects created by the SQL/400 licensed program that consists of and logically classifies a set of objects, such as tables, views, and indexes.

command. (1) A request for performance of an operation or execution of a program. (2) A character string from a source external to a system that represents a request for system action.

comment. A set of characters enclosed in double quotation marks. Smalltalk ignores comments and does not execute them.

Common Gateway Interface. A standard protocol through which a Web server can execute programs running on the server machine. CGI programs are executed in response to requests from Web client browsers.

Common Object Request Broker Architecture (CORBA). An architectural standard proposed by the Object Management Group (OMG), an industry standards organization, for creating object descriptions that are portable among programming languages and execution platforms.

Common Programming Interface Communications (CPI-C). An IBM communications architecture that defines a programming interface for peer-to-peer communications that is common across different environments and platforms. Also referred to as CPI Communications or CPI-C.

Common User Access (CUA). An IBM architecture for designing graphical user interfaces that uses a set of standard components and terminology.

component. A functional grouping of classes and related files within a product. See also *system component*.

composite part. A part that contains other parts; it can also contain data and behavior of its own. For example, a user interface view is a composite part composed of subparts such as entry fields, push buttons, and text.

Composition Editor. In VisualAge, a view that is used to build a graphical user interface and to make connections among parts.

concrete class. A subclass of an abstract class that is a specialization of the abstract class. For example, the concrete class, OrderedCollection, is a subclass of the abstract class, Collection.

configuration. (1) A description of a group of components that identifies, for each component, the component edition or version that is part of the group. (2) The arrangement of a computer system or network as defined by the nature, number, and chief characteristics of its functional units. More specifically, the term may refer to a hardware configuration or a software configuration. (3) The devices and programs that make up a system, subsystem, or network.

configuration file. The collective set of definitions that describes a configuration.

configuration map. A named group of application editions. A configuration map usually represents a product or one of its major parts.

connection. (1) In VisualAge, a formal, explicit relationship between parts. Connections define the ways in which parts communicate with one another. Making connections is the basic technique used for building any VisualAge application. See also *attribute-to-script*, *attribute-to-attribute*, *event-to-script*, and *event-to-action* connection. (2) A linkage between nodes. Connections are established and released at the Network, Session, and Presentation Layers.

construction from parts. A software development technology in which applications are assembled from reusable and existing software components known as parts.

control language. The set of all commands with which users request functions from the system.

controller. A unit that controls input/output operations for one or more devices.

conversation. In SNA, a logical connection between two transaction programs using an LU 6.2 session. Conversations are delimited by brackets to gain exclusive use of a session.

CORBA. See *Common Object Request Broker Architecture*.

CPI-C. See *Common Programming Interface Communications*.

CUA. See *Common User Access*.

Customer Information Control System (CICS). An IBM licensed program that enables transactions entered at remote terminals to be processed by user-written applications. It includes facilities for building, using, and maintaining databases.

Cyberspace. Term originated by author William Gibson in his novel *Neuromancer*, the word Cyberspace is currently used to describe the whole range of information resources available through computer networks.

D

data area. A system object used to communicate data such as common language variable values between the programs within a job and between jobs. A data area is identified to the system as a specific object type. The system-recognized identifier for the object type is *DTAARA.

database file. A system object of the type *FILE that contains descriptions of how input data is to be presented to a program from internal storage and how output data is to be presented to internal storage from a program. The collection of all database files makes up the database (all the data files stored in the system).

database manager. A VisualAge or IBM Smalltalk database component that models a database management system in order to provide the interface between an application and the database management system.

data description specifications (DDS). A format for describing the user′s database files or device files to the system. Describing a file in DDS is similar to filling in information on a form that is arranged in columns and rows. The most common characteristics, such as the names and lengths of fields, are described by putting entries in specific columns on the form. Another part of the form allows special parameters that describe less common and more varied characteristics. The finished specifications are then used as the source for creating the file.

data integrity. (1) The condition that exists as long as accidental or intentional destruction, alteration, or loss of data does not occur. (2) Preservation of data for its intended use.

data processing. The systematic performance of operations upon data; for example, handling, merging, sorting, and computing.

data queue. A way to communicate and put data used by several programs in a job or between jobs. The data queue is identified to the system as a specific type of object. The system-recognized identifier for the object type is *DTAQ.

data structure. The syntactic structure of symbolic expressions and their storage allocation characteristics.

data transfer. (1) The result of the transmission of data signals from any data source to a data receiver. (2) The movement, or copying, of data from one location and the storage of the data at another location.

deactivate. To take a resource of a node out of service, rendering it inoperable, or to place it in a state in which it cannot perform the functions for which it was designed. Contrast with *activate*.

deallocate. A logical unit (LU) 6.2 application program interface (API) verb that terminates a conversation, thereby freeing the session for a future conversation. Contrast with *allocate*.

debugger. A software tool used to detect, trace, and eliminate errors in computer programs or other software.

default. Pertaining to an attribute, value, or option that is assumed when none is explicitly specified.

delimiter. (1) A character used to indicate the beginning or end of a character string. (2) A bit pattern that defines the beginning or end of a frame or token on a LAN.

dependent LU. Any logical unit (LU) that receives an ACTLU over a link. Such LUs can act only as secondary logical units (SLUs) and can have only one LU-LU session at a time. Contrast with *independent LU*.

destination. Any point or location, such as a node, station, or particular terminal, to which information is to be sent.

device. An input/output unit such as a terminal, display, or printer.

dictionary. In Smalltalk, an unordered collection whose elements are accessed by an explicitly assigned external key. See also *pool dictionary*.

digital. (1) Pertaining to data in the form of digits. Contrast with *analog*. (2) Pertaining to data consisting of numerical values or discrete units.

disabled. (1) Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. (2) Pertaining to the state in which a transmission control unit or audio response unit cannot accept incoming calls on a line.

display. (1) To present information for viewing, usually on a terminal screen or a hard-copy device. (2) A device or medium on which information is presented, such as a terminal screen.

Display. (1) A Smalltalk command that executes the selected code and displays the result. (2) In IBM Smalltalk, an X/Motif concept that models the user′s hardware display. The functions of the X/Motif Display object are implemented in the IBM Smalltalk CgDisplay class.

distributed application. A workstation application that runs in cooperation with programs running on other processes or machines. Client/server applications are a subset of distributed applications.

distributed computing environment (DCE). A set of services and tools that support the creation, use, and maintenance of distributed applications in a heterogeneous computing environment.

Distributed System Object Model (DSOM). An extension to SOM enabling SOM objects to reside on multiple network nodes.

DLL. See *dynamic link library*.

domain. (1) An access method, its application programs, communication controllers, connecting lines, modems, and attached terminals. (2) In SNA, a system services control point (SSCP) and the physical units (PUs), logical units (LUs), links, link stations, and all the associated resources that the SSCP has the ability to control by means of activation requests and deactivation requests.

domain name. The unique name that identifies an Internet site. Domain names always have two or more parts, separated by dots. The part on the left is the most specific, and the part on the right is the most general. A given machine may have more than one domain name but a given domain name points to only one machine. Usually, all of the machines on a given network will have the same thing as the right-hand portion of their domain names, for example, gateway.mynetwork.com.br, mail.mynetwork.com.br, www.mynetwork.com.br, and so on. It is also possible for a domain name to exist but not be connected to an actual machine. This is often done so that a group or business can have an Internet e-mail address without having to establish a real Internet site. In these cases, some real Internet machine must handle the mail on behalf of the listed domain name. See also IP Number.

dynamic data exchange (DDE). A communication mechanism between processes that enables two applications to exchange data in a client/server relationship.

dynamic link library (DLL). A file containing data and code objects that can be used by programs or applications during loading or at run time but are not part of the program′s executable (.EXE) file.

E

EBCDIC. Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

edition. In the VisualAge or IBM Smalltalk team programming environment, a software component that is subject to further change. A software component can have one or more editions, identified by a time stamp stating the date and time of the edition′s creation. Many changes can be made to a single edition of a class. In contrast, every change to a method creates a new edition of that method. In its broadest sense, edition can include scratch edition and version.

EHLLAPI. Emulator high-level language application programming interface. A programming interface that enables a workstation application to communicate with a mainframe application. EHLLAPI operates in conjunction with a terminal (such as 3270) emulator.

e-mail. (Electronic mail) Messages transmitted over the Internet from user to user. E-mail can contain text, but also can carry with it files of any type as attachments.

enabled. (1) On a LAN, pertaining to an adapter or device that is active, operational, and able to receive frames from the network. (2) Pertaining to a state of a processing unit that allows the occurrence of certain types of interruptions. (3) Pertaining to the state in which a transmission control unit or an audio response unit can accept incoming calls on a line.

encapsulation. The hiding of a software object′s internal data representation. The object provides an interface that queries and manipulates the data without exposing its underlying structure.

end user. A person, device, program, or computer system that utilizes a computer network for the purpose of data processing and information exchange.

enterprise. A business or organization that consists of two or more sites separated by a public right-of-way or a geographical distance.

Ethernet. A very common method of networking computers in a *LAN*. Ethernet will handle about 10,000,000 bits/second and can be used with almost any kind of computer. See also bandwidth, LAN.

event. A representation of a change that occurs to a part. The events on a part′s public interface enable other interested parts to receive notification when something about the part changes. For example, a push button generates an event signaling that it has been clicked, which might cause another part to display a window. Compare to *attribute*.

event-to-action connection. A connection that causes an action to be performed when an event occurs. See also *connection*.

event-to-script connection. A connection that causes a script to run when an event occurs. See also *connection*.

exception. An abnormal condition such as an I/O error encountered in processing a data set or a file.

execute. To perform the actions specified by a program or a portion of a program.

execution. The process of carrying out an instruction or instructions of a computer program by a computer.

expression. In Smalltalk, the syntactic representation of one or more messages. An expression can consist of subexpressions representing the receiver and arguments of the message. The expression can also cause the assignment of its result to one or more variables.

external source. The format of Smalltalk source code that is filed out to an external file. See also *file in* and *file out*.

F

feature. (1) A major component of a software product that can be ordered separately. (2) In VisualAge, an action, attribute, or event that is available from a part's public interface and to which other parts can connect. See also *attribute* and *event*.

field. A group of related bytes (such as name or amount) that are treated as a unit in a record.

file. (1) A generic term for the object type that refers to a database file, a device file, or a save file. The system-recognized identifier for the object type is *FILE. (2) In the hierarchical file system, a piece of related information (data), such as a document. (3) In SQL, the term is generally referred to as a table.

file in. A Smalltalk command for compiling external definitions of applications, classes, and methods from a text file.

file name. (1) A name assigned to or declared for a file. (2) The name used by a program to identify a file.

file out. A Smalltalk command for writing definitions of applications, classes, and methods to an external text file.

firewall. A combination of hardware and software that protects a local area network (LAN) from Internet hackers. It separates the network into two or more parts and restricts outsiders to the area outside the firewall. Private or sensitive information is kept inside the firewall.

first-in first-out (FIFO). A queuing technique in which the next request to be processed from a queue is the request of the highest priority that has been on the queue for the longest time.

fixed-length record. A record having the same length as all other records with which it is logically or physically associated.

form. An HTML element that can include entry fields, push buttons, and other user-interface controls through which users can enter information. Sometimes called a *fill-in form*.

format. (1) A specified arrangement of such things as characters, fields, and lines, usually used for displays, printouts, or files. (2) To arrange such things as characters, fields, and lines.

FQDN. (Fully Qualified Domain Name) The official name assigned to a computer. Organizations register names, such as *ibm.com* or *utulsa.edu*. They then assign unique names to their computers, such as *watson5.ibm.com* or *tornado.cs.utulsa.edu*.

framework. A library of classes, intended for reuse, that fall within a particular domain (for example, a communications framework or a graphics framework).

free-form surface. In VisualAge, the large open area of the Composition Editor window. The free-form surface holds the visual parts contained by the views a user builds and representations of the nonvisual parts that an application includes.

FTP. (File Transfer Protocol) The basic Internet function that enables files to be transferred between computers. You can use it to download files from a remote, host computer, as well as to upload files from your computer to a remote, host computer. (See Anonymous FTP).

function. (1) A specific purpose of an entity, or its characteristic action. (2) In data communications, a machine action such as carriage return or line feed.

G

garbage collection. A Smalltalk process for periodically identifying unreferenced objects and deallocating their memory.

gateway. A device and its associated software that interconnect networks or systems of

different architectures. The connection is usually made above the reference model network layer.

GET. One of the methods used in HTTP requests. A GET request is used to retrieve data from an HTTP server. See also POST.

GIF. (Graphics Interchange Format) A graphics file format that is commonly used on the Internet to provide graphics images in Web pages.

global variable. A variable that any method in any object can access.

graphical user interface (GUI). A type of interface that enables users to communicate with a program by manipulating graphical elements rather than by entering commands. Typically, a graphical user interface includes a combination of graphics, pointing devices, menu bars, overlapping windows, and icons.

graphics. (1) The making of charts and pictures. (2) Pertaining to charts, tables, and their creation.

group member. A team member who belongs to a group that is responsible for developing an application.

GUI. See *graphical user interface*.

H

hardware. Physical equipment as opposed to programs, procedures, rules, and associated documentation.

header. The portion of a message that contains control information for the message such as one or more destination fields, name of the originating station, input sequence number, character string indicating the type of message, and priority level for the message.

HLLAPI. High-level language application programming interface. A programming interface that enables a workstation application to communicate with a mainframe application. HLLAPI usually operates in conjunction with a terminal emulator.

host. (1) A computer that ″hosts″ outside computer users by providing files, services or sharing its resources. (2) Any computer on a network that is a repository for services available to other computers on the network. It is quite common to have one host machine provide several services, such as *WWW* and *USENET*. See also Node, Network.

host computer. (1) The primary or controlling computer in a multicomputer installation or network. (2) In a network, a processing unit in where a network access method resides.

host variable. A variable in an SQL statement used for substituting data values into the statement at execution time.

HTML (hypertext markup language).. The basic language that is used to build hypertext documents on the World Wide Web. It is used in basic, plain ASCII-text documents, but when those documents are interpreted (called rendering) by a Web browser such as Netscape, the document can display formatted text, color, a variety of fonts, graphic images, special effects, hypertext jumps to other Internet locations and information forms.

HTTP (hypertext transfer protocol). The protocol for moving hypertext files across the Internet. Requires a HTTP client program on one end, and an HTTP server program on the other end. HTTP is the most important protocol used in the World Wide Web (WWW). See also Client, Server, WWW.

HTTP request. A transaction initiated by a Web browser and adhering to HTTP. The server usually responds with HTML data, but can send other kinds of objects as well.

hypertext. Text in a document that contains a hidden link to other text. You can click a mouse on a hypertext word and it will take you to the text designated in the link. Hypertext is used in Windows help programs and CD encyclopedias to jump to related references elsewhere within

the same document. The wonderful thing about hypertext, however, is its ability to link– using HTTP over the Web – to any Web document in the world, yet still require only a single mouse click to jump clear around the world.

I

icon. A small pictorial representation of an object.

image. A Smalltalk file that provides a development environment on an individual workstation. An image contains object instances, classes, and methods. It must be loaded into the Smalltalk virtual machine in order to run.

implementor. For any given message selector, a method that implements that selector.

IMS. Information Management System/Virtual Storage.

inactive. (1) Not operational. (2) Pertaining to a node or device not connected or not available for connection to another node or device.

independent LU. A logical unit (LU) that does not receive an ACTLU over a link. Such LUs can act as primary logical units (PLUs) or secondary logical units (SLUs) and can have one or more LU-LU sessions at a time. Contrast with *dependent LU*.

index. A set of pointers that are logically arranged by the values of a key. Indexes provide quick access and can enforce uniqueness on the rows in a table.

inheritance. A relationship among classes in which one class shares the structure and behavior of another. A subclass inherits from a superclass.

initialize. In a LAN, to prepare the adapter (and adapter support code, if used) for use by an application program.

input/output (I/O). (1) Pertaining to a device whose parts can perform an input process and an output process at the same time. (2) Pertaining to a functional unit or channel involved in an input process, output process, or both, concurrently or not, and to the data involved in such a process.

inspector. A Smalltalk tool for viewing the data of any Smalltalk object.

instance. An object that is a single occurrence of a particular class. An instance exists in memory or external media in persistent form. Compare to persistent object.

instance method. In Smalltalk, a method that provides behavior for particular instances of a class. Messages that invoke instance methods are sent to particular instances, rather than to the class as a whole. Contrast with *class method*.

instance variable. Private data that belongs to an instance of a class and is hidden from direct access by all other objects. Instance variables can be accessed only by the instance methods of the defining class and its subclasses.

interface. (1) A shared boundary between two functional units, defined by functional characteristics, common physical interconnection characteristics, signal characteristics, and other characteristics as appropriate. (2) A shared boundary. An interface may be a hardware component to link two devices or a portion of storage or registers accessed by two or more computer programs. (3) Hardware, software, or both, that links systems, programs, or devices.

interface aids. In VisualAge, when used together with the RecordStructure classes, the interface aids convert existing C or COBOL definitions into Smalltalk objects.

Internet. The vast collection of interconnected networks that all use the TCP/IP protocols and that evolved from the ARPANET of the late 1960′s and early 1970′s. By July of 1995, the Internet was connecting roughly 60,000 independent networks into a vast global net.

intranet. A private *network* inside a company or organization that uses the same kinds of software that you would find on the public *Internet*, but that is only for internal use. As the Internet has become more popular, many of the tools used on the Internet are being used in private networks, for example, many companies have Web servers that are available only to employees.

IP. (Internet Protocol) The rules that provide basic Internet functions. See TCP/IP.

IP Number. An Internet address that is a unique number consisting of four parts separated by dots, sometimes called a *dotted quad*. (For example: 198.204.112.1) Every Internet computer has an IP number and most computers also have one or more domain names that are plain language substitutes for the dotted quad.

ISDN. (Integrated Services Digital Network) A set of communications standards that enable a single phone line or optical cable to carry voice, digital network services and video. ISDN is intended to eventually replace our standard telephone system.

iterative development. A software development process that allows progress in stages. At the end of each stage, the result is verified by end users. Through such verification, requirements are dynamically identified and refined while the product is under development.

J

Java. Java is a new programming language invented by Sun Microsystems that is specifically designed for writing programs that can be safely downloaded to your computer through the Internet and immediately run without fear of viruses or other harm to your computer or files. Using small Java programs (called *applets*, Web pages can include functions such as animations, calculators, and other fancy tricks. We can expect to see a huge variety of features added to the Web using Java, since you can write a Java program to do almost anything a regular computer program can do, and then include that Java program in a Web page.

job control language (JCL). A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

journal. A system object that identifies the objects being journaled, the current journal receiver, and all journal receivers on the system for the journal. The system-recognized identifier for the object type is *JRN.

journaling. The process of recording, in a journal, the changes made to a physical file member. Journaling allows the programmer to reconstruct a physical member by applying or removing the changes in the journal to a saved version of the physical file member.

journal receiver. A system object that contains journal entries added when changes are made to an object, for example, when an update is made to a file being journaled. The system-recognized identifier for the object type is *JRNRCV.

JPEG. (Joint Photographic Experts Group) The name of the committee that designed the photographic image-compression standard. JPEG is optimized for compressing full-color or gray-scale photographic-type, digital images. It doesn't work well on drawn images such as line drawings, and it does not handle black-and-white images or video images.

K

kbps. (kilobits per second) A speed rating for computer modems that measures (in units of 1024 bits) the maximum number of bits the device can transfer in one second under ideal conditions.

keyword. (1) One of the predefined words of an artificial language. (2) One of the significant and informative words in a title or document that describes the content of that document.

(3) A name or symbol that identifies a parameter. (4) A part of a command operand that consists of a specific character string (such as DSNAME=).

keyword message. A message that takes one or more arguments. A keyword is an identifier followed by a colon (:). Each keyword requires one argument, and the order of the keywords is important. ′hello′ at: 2 put: $H is an example of a keyword message; at: and put: are keyword selectors, 2 and $H are the arguments. See also *message.*

kilobyte. A thousand bytes. Actually, usually 1024 (2^10) bytes. See also byte, bit.

L

LAN. See *local area network.*

library. (1) A shared repository represented by a single file. It stores source code, object (compiled) code, and persistent objects, including editions, versions, and releases of software components. (2) A system object that serves as a directory to other objects. A library groups related objects and allows the user to find objects by name. To identify a specific object to the system, a user needs only to provide the object name, the object type, and the name of the library containing the object.

link. (1) The logical connection between nodes including the end-to-end link control procedures. (2) The combination of physical media, protocols, and programming that connects devices on a network. (3) In computer programming, the part of a program, in some cases a single instruction or an address, that passes control and parameters between separate portions of the computer program. (4) To interconnect items of data or portions of one or more computer programs. (5) In SNA, the combination of the link connection and link stations joining network nodes.

listserv. An Internet application that automatically serves mailing lists by sending electronic newsletters to a stored database of Internet user addresses. Users can handle their own subscribe/unsubscribe actions without requiring anyone at the server location to personally handle the transaction.

literal. An object that can be created by the compiler. A literal can be a number, a character string, a single character, a symbol, or an array. All literals are unique: Two literals with the same value refer to the same object. The object created by a literal is read-only; it cannot be changed.

literal text. Text in an HTML Text part that is passed to the client browser exactly as entered. You can use literal text to code HTML tagging that is not directly supported by the Web Connection parts.

load. A system operation that links the compiled code for a software component from a library into an active image. Loading also performs other operations that enable the component to run, such as linking prerequisites.

local address. In SNA, an address used in a peripheral node in place of an SNA network address and transformed to or from an SNA network address by the boundary function in a subarea node.

local area network (LAN). (1) A network in which devices are connected to one another for communication and can be connected to a larger network. See also *token ring*. (2) A network in which communications are limited to a moderately sized geographic area such as a single office building, warehouse, or campus, and do not generally extend across public rights-of-way. Contrast with *wide area network*.

local node. In the subsystem, the node from which one views the rest of the OSI network—the node for which resources are defined. Contrast with *remote node*.

logical file. A description of how data is to be presented to or received from a program. This type of database file contains no data, but it defines record formats for one or more physical files. The record formats allow a developer to

present different views of the data in a physical file.

logical unit (LU). In SNA, a port through which a user gains access to the services of a network. A logical unit can support two types of sessions—with the host, and with other LUs. See *logical unit 6.2*.

logical unit (LU) 6.2. A type of logical unit that supports general communication between programs in a distributed processing environment. LU 6.2 is characterized by (1) a peer relationship between session partners, (2) efficient utilization of a session for multiple transactions, (3) comprehensive end-to-end error processing, and (4) a generic application program interface (API) consisting of structured verbs that are mapped to a product implementation. Synonym for *Advanced Program-to-Program Communications*.

Login. The account name used to gain access to a computer system. Not kept secret (unlike password).

LU type. In SNA, the classification of a LU-LU session in terms of the specific subset of SNA protocols and options supported by the logical units (LUs) for that session, namely:

- The mandatory and optional values allowed in the session activation request
- The usage of data stream controls, function management headers (FMHs), request unit (RU) parameters, and sense codes
- Presentation services protocols such as those associated with FMH usage.

LU types 0, 1, 2, 3, 4, 6.1, 6.2, and 7 are defined.

M

machine-readable information (MRI). Language-sensitive information associated with a computer program, such as program integrated information or softcopy documentation.

management services. In SNA, one of the types of network services in control points (CPs) and physical units (PUs). Management services are the services provided to assist in the management of SNA networks, such as problem management, performance and accounting management, configuration management and change management.

megabyte. A million bytes. A thousand kilobytes. See also byte, bit, kilobyte.

message. In Smalltalk, a communication from one object to another that requests the receiving object to execute a method. A message consists of a reference to the receiving object, followed by a selector indicating the requested method, and (in many cases) arguments to be used in executing the method. There are three types of messages: binary, keyword, and unary.

metaclass. The specification of a class; the complete description of a class′s attributes, behavior, and implementation. Every class has a metaclass, of which it is the sole instance. Contrast with *class*.

method. Executable code that implements the logic of a particular message for a class. In VisualAge, methods are also called scripts. See also *class method*, *instance method*, *private method*, and *public method*.

MIME. (Multipurpose Internet Mail Extensions) A set of Internet functions that extend normal e-mail capabilities and enable nontext computer files to be attached to e-mail. Nontext files include graphics, spreadsheets, formatted word-processor documents, sound files, and so on. Files sent by MIME arrive at their destination as exact copies of the original so that you can send fully formatted word processing files, spreadsheets, graphics images and software applications to other users via simple e-mail. Besides email software, the MIME standard is also universally used by Web servers to identify the files they are sending to Web clients, in this way new file formats can be accommodated simply by updating the browsers′ list of pairs of MIME types and

appropriate software for handling each type. See also browser, client, server.

model. A nonvisual part that represents the state and behavior of a real-world object, such as a customer or an account. Contrast with view.

mode name. A symbolic name for a set of session characteristics. For LU 6.2, a mode name and a partner LU name together define a group of parallel sessions having the same characteristics.

module. A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to or output from an assembler, compiler, linkage editor, or executive routine.

monitor. (1) A functional unit that observes and records selected activities for analysis within a data processing system. Possible uses are to show significant departures from the norm, or to determine levels of utilization of particular functional units. (2) Software or hardware that observes, supervises, controls, or verifies operations of a system.

Multiple Virtual Storage (MVS). An IBM licensed program whose full name is the Operating System/Virtual Storage (OS/VS) with Multiple Virtual Storage/System Product for System/370. It is a software operating system controlling the execution of programs.

multireceive dialog. A dialog that takes one data record from the object that initiates the dialog, sends it to a remote program, and returns multiple records from the remote program to the initiating object. Compare to *simple dialog*.

N

NetBIOS. See *Network Basic Input/Output System*.

network. (1) A configuration of data processing devices and software connected for information interchange. (2) An arrangement of nodes and connecting branches. Connections are made between data stations.

network address. In SNA, an address, consisting of subarea and element fields, that identifies a link, a link station, or a network addressable unit. Subarea nodes use network addresses; peripheral nodes use local addresses. The boundary function in the subarea node to which a peripheral node is attached transforms local addresses to network addresses and vice versa. See *local address*.

Network Basic Input/Output System (NetBIOS). An operating system interface for application programs used on IBM personal computers that are attached to an IBM Token-Ring Network.

network layer. (1) In the Open Systems Interconnection reference model, the layer that provides for the entities in the transport layer the means for routing and switching blocks of data through the network between the open systems in which those entities reside. (2) The layer that provides services to establish a path between systems with a predictable quality of service. See *Open Systems Interconnection (OSI)*.

network management. The conceptual control element of a station that interfaces with all of the architectural layers of that station and is responsible for the resetting and setting of control parameters, obtaining reports of error conditions, and determining whether the station should be connected to or disconnected from the network.

nil. The object in Smalltalk that means *no value*. All variables initially refer to nil. It is the single instance of the UndefinedObject class.

node. (1) Any device, attached to a network, that transmits and/or receives data. (2) An endpoint of a link, or a junction common to two or more links in a network. (3) In a network, a point where one or more functional units interconnect transmission lines.

node name. In VTAM, the symbolic name assigned to a specific major or minor node during network definition.

nonvisual class. A class in a VisualAge application that specifies a nonvisual part. For example, Person, Address, and BankAccount are nonvisual classes.

nonvisual part. A part that has no visual representation at run time. A nonvisual part typically represents some real-world object that exists in the business environment. Contrast with *view, visual part.*

notebook. In VisualAge, a view that resembles a bound notebook, containing pages separated into sections by tabbed divider pages. A user can turn the pages of a notebook or select the tabs to move from one section to another.

O

object. (1) The basic building block in Smalltalk development. An object is anything that exhibits behavior. All code and data in Smalltalk must be part of an object. (2) On the AS/400 system, an object is a named storage space consisting of a set of characteristics that describes itself and, in some cases, data. Some examples of objects are programs, files, and libraries.

object code. Compiler or assembler output that is itself executable machine code or is suitable for processing to produce executable machine code. Contrast with *source code*.

object factory. A nonvisual part capable of dynamically creating new instances of a specified part. For example, during the execution of an application, an object factory can create instances of a new class to collect the data being generated.

object-oriented programming. A programming methodology built around objects and based on sending messages back and forth between those objects. The basic concepts of object-oriented programming are encapsulation, inheritance, and polymorphism.

object persistency. A characteristic that enables objects to exist beyond the time in which their creating application runs. One use of object persistency is sharing objects among programmers in a development environment.

open system. (1) A system with specified standards that therefore can be readily connected to other systems that comply with the same standards. (2) A data communications system that conforms to the standards and protocols defined by Open Systems Interconnection (OSI).

open systems architecture (OSA). A model that represents a network as a hierarchical structure of layers of functions; each layer provides a set of functions that can be accessed and that can be used by the layer above it.

Note: Layers are independent in the sense that implementation of a layer can be changed without affecting other layers.

Open Systems Interconnection (OSI). (1) The interconnection of open systems in accordance with specific ISO standards. (2) The use of standardized procedures to enable the interconnection of data processing systems. **Note:** The OSI architecture establishes a framework for coordinating the development of current and future standards for the interconnection of computer systems. Network functions are divided into seven layers. Each layer represents a group of related data processing and communication functions that can be carried out in a standard way to support different applications.

operating system. Software that controls the execution of programs. An operating system can provide services such as resource

allocation, scheduling, input/output control, and data management.

operation. (1) A defined action, namely, the act of obtaining a result from one or more operands in accordance with a rule that completely specifies the result for any permissible combination of operands. (2) A program step undertaken or executed by a computer. (3) An action performed on one or more data items, such as adding, multiplying, comparing, or moving.

option. (1) A specification in a statement, a selection from a menu, or a setting of a switch that may be used to influence the execution of a program. (2) A hardware or software function that may be selected or enabled as part of a configuration process. (3) A piece of hardware (such as a network adapter) that can be installed in a device to modify or enhance device function.

P

panel. (1) A formatted display of information that appears on a terminal screen. Contrast with *screen*. (2) In computer graphics, a display image that defines the locations and characteristics of display fields on a display surface.

parameter. (1) A data element included as part of a message to provide information that the object might need. In Smalltalk, generally referred to as an argument. (2) A variable that is given a constant value for a specified application and that may denote the application. (3) An item in a menu or for which the user specifies a value or for which the system provides a value when the menu is interpreted. (4) Data passed between programs or procedures.

part. A self-contained software object with a standardized public interface, consisting of a set of external features that allow the part to interact with other parts. The parts on the VisualAge parts palette can be used as templates to create instances of objects.

parts palette. In the VisualAge Composition Editor, an organized collection of visual and nonvisual parts used in building composite parts for an application. The parts palette is organized into categories. Application developers can add parts to the palette for use in defining applications or other parts.

password. In computer security, a string of characters known to the computer system and a user, who must specify it to gain full or limited access to a system and to the data stored within it.

path. (1) In a network, any route between any two nodes. (2) The route traversed by the information exchanged between two attaching devices in a network. (3) A command in IBM Personal Computer Disk Operating System (PC DOS) and IBM Operating System/2 (OS/2) that specifies directories to be searched for commands or batch files that are not found by a search of the current directory.

PATH_INFO. A CGI variable, usually transmitted to the CGI program in the form of an environment variable. The PATH_INFO variable contains all path information from the URL following the name of the CGI executable. For a Web Connection application, this information is the same as the VisualAge part name.

persistent object. Instances stored outside of the image. A persistent object must be loaded into virtual or real storage before it can process messages sent to it.

personal computer (PC). A desk-top, free-standing, or portable microcomputer that usually consists of a system unit, a display, a monitor, a keyboard, one or more diskette drives, internal fixed-disk storage, and an optional printer. PCs are designed primarily to give independent computing power to a single user and are inexpensively priced for purchase by individuals or small businesses. Examples include the various models of the IBM personal computers and the IBM Personal System/2 computer.

physical connection. The ability of two connectors to mate and make electrical contact. In a network, devices that are physically connected can communicate only if they share the same protocol.

physical file. A description of how data is to be presented to or received from a program and how data is actually stored in the database. A physical file contains one record format and one or more members.

physical unit (PU). In SNA, a type of network addressable unit (NAU). A physical unit (PU) manages and monitors the resources (such as attached links) of a node, as requested by a system services control point (SSCP) through an SSCP-PU session. An SSCP activates a session with the physical unit in order to indirectly manage, through the PU, resources of the node such as attached links.

pointer. (1) An identifier that indicates the location of an item of data. (2) A data element that indicates the location of another data element. (3) A physical or symbolic identifier of a unique target.

polymorphism. The ability of different objects to respond to the same message in different ways. Different objects can have very different method implementations for the same message. An object can send a message without concern for its underlying implementation.

pool dictionary. A dictionary object whose keys define variables that can be shared by multiple classes. All methods for a class can access the variables in a pool dictionary if the class declares the pool dictionary as part of its scope.

port. (1) A place where information goes into or out of a computer, or both. For example, the serial port on a personal computer is where a modem would be connected. (2) On the Internet port often refers to a number that is part of a URL, appearing after a colon (:) right after the domain name. Every *service* on an Internet server listens on a particular port number on that server. Most services have standard port numbers; Web servers normally listen on port 80. Services can also listen on nonstandard ports, in which case the port number must be specified in a URL when accessing the server. (3) Refers to translating a piece of software to bring it from one type of computer system to another. See also domain name, server, URL.

POST. One of the methods used in HTTP requests. A POST request is used to send data to an HTTP server. See also GET.

prerequisite application. An application required by another application for it to function successfully. An application can extend or reference one or more of the prerequisite application′s classes. In team development, prerequisite applications are particular versions or editions of applications.

presentation layer. In the Open Systems Interconnection reference model, the layer that provides for the selection of a common syntax for representing information and for transformation of application data into or from this common syntax.

primary logical unit (PLU). In SNA, the logical unit (LU) that contains the primary half-session for a particular LU-LU session. Each session must have a PLU and secondary logical unit (SLU). The PLU is the unit responsible for the bind and is the controlling LU for the session. A particular LU may contain both primary and secondary half-sessions for different active LU-LU sessions.

primary part. In a composite part constructed with the VisualAge Composition Editor, the subpart whose public interface is fully exposed on the public interface of the composite part. The primary part is transparently visible to parts outside the composite part and is the subpart with which most interaction will take place.

private class. In VisualAge or IBM Smalltalk, a class that is not part of the system API but is provided as part of the internal functioning of the system. A private class is not visible outside its containing application. If you use a VisualAge or IBM Smalltalk class marked as private, you might have to modify your code in

order for it to work with a future release of VisualAge or IBM Smalltalk. Contrast with *public class*.

private method. In VisualAge or IBM Smalltalk, a method that is not part of the system API but is provided as part of the internal functioning of the system. If you use a method marked as private, you might have to modify your code in order for it to work with a future release of VisualAge or IBM Smalltalk. Contrast with *public method*. Individual application development projects can use the public and private designations as a means of organizing their code.

process. In Smalltalk, a sequence of actions described by expressions and performed by the system′s virtual machine.

property. A unique characteristic of a part.

protocol. (1) The set of all messages to which an object will respond. (2) Specification of the structure and meaning (the semantics) of messages that are exchanged between a client and a server. (3) A set of semantic and syntactic rules that determine the behavior of functional units in achieving communication. (4) In SNA, the meanings of and the sequencing rules for requests and responses used for managing the network, transferring data, and synchronizing the states of network components. (5) A specification for the format and relative timing of information exchanged between communicating parties.

proxy. An application gateway from one network to another for a specific network application like Telnet of FTP, for example, a firewall′s proxy Telnet server performs authentication of the user and then lets the traffic flow through the proxy as if it were not there. Function is performed in the firewall and not in the client workstation, causing more load in the firewall. Compare with socks.

PRPQ. Programming Request for Price Quotation. A customer request for a price quotation for a licensed program to be designed especially for a particular group of customers or an application. Documentation for the program is provided only to those customers who order the PRPQ.

pseudocode. An artificial language used to describe computer program algorithms without using the syntax of any particular programming language. The code requires translation before execution.

public class. A class that is provided as part of the VisualAge or IBM Smalltalk API. A public class is visible to applications other than its containing application. Public classes are designed to function with future releases of VisualAge or IBM Smalltalk and with operating systems supported by VisualAge or IBM Smalltalk. Contrast with *private class*.

public interface. A set of external features that enable a part to interact with other parts. A part′s public interface is made up of three characteristics: attributes, actions, and events.

Public Interface Editor. A VisualAge view used to create and modify attributes, actions, and events, which together make up the public interface of a part.

public method. A method that is provided as part of the VisualAge or IBM Smalltalk API. Public methods are designed to function with future releases of VisualAge or IBM Smalltalk and with operating systems supported by VisualAge or IBM Smalltalk. Contrast with *private method*. Individual application development projects can use the public and private designations as a means of organizing their code.

Q

query specification. A database query definition. All queries issued to a database by VisualAge or IBM Smalltalk must be defined by a query specification.

quick form. In the VisualAge Composition Editor, a menu option that enables application

developers to quickly create a default view for a part.

R

receive. To obtain and store information transmitted from a device.

receiver. The object that receives a message. Contrast with *sender*.

record. (1) (ISO) In programming languages, an aggregate that consists of data objects, possibly with different attributes, that usually have identifiers attached to them. In some programming languages, records are called structures. (2) (TC97) A set of data treated as a unit. (3) A set of one or more related data items grouped for processing. (4) In VTAM, the unit of data transmission for record mode. A record represents whatever amount of data the transmitting node chooses to send.

RecordStructure. An object that contains information about the format, structure, and types of the data it contains.

RecordStructure classes. Classes that provide a flexible, open architecture for converting Smalltalk objects to and from data types of other languages.

release. A system operation on a component that changes its containing component′s configuration. Releasing a component adds its released edition or version to the configuration for its containing component. When a containing component is loaded into an image, the released editions or versions of the components it contains are also loaded.

remote. Concerning the peripheral parts of a network not centrally linked to the host processor and generally using telecommunication lines with public right-of-way.

remote node. Any node other than the local node. Contrast with *local node*.

repository. (1) An organized, shared body of information that can support business and data-processing activities. (2) In VisualAge or IBM Smalltalk, the multiuser library that stores components such as applications, classes, and methods created by application developers. It stores source code, object code, and persistent objects.

request. A service primitive issued by a service user to call a function supported by the service provider.

reset button. A type of push button that can appear on a form. A reset button restores all input fields to their default states.

resource. (1) Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In the NetView program, any hardware or software that provides function to the network.

response. A service primitive issued by a service user to complete the procedures associated with a confirmed service.

return code. (1) A value (usually hexadecimal) provided by an adapter or a program to indicate the result of an action, command, or operation. (2) A code used to influence the execution of succeeding instructions.

return value. An object or data type that a receiver object passes to a sender object in response to a message.

router. A network device that enables the network to reroute messages it receives that are intended for other networks. The network with the router receives the message and sends it on its way exactly as received. In normal operations, they do not store any of the messages that they pass through.

routine. Part of a program, or a sequence of instructions called by a program, that may have some general or frequent use.

RPG. A programming language designed for writing application programs for business data processing requirements. The application programs range from report writing and inquiry programs to applications such as payroll, order entry, and production planning.

S

scratch edition. A mutable and private copy of an application for a user who is not necessarily the application′s manager. The scratch edition only exists in that user′s image. Using a scratch edition, one can modify an application version, and the existing classes contained in it, without actually creating a new edition. Each scratch edition has << >> displayed around the edition timestamp or version name. Contrast with edition, version.

screen. An illuminated display surface; for example, the display surface of a CRT or plasma panel. Contrast with *panel*.

script. A series of Smalltalk statements that implement an action for a part. Scripts are equivalent to Smalltalk methods.

Script Editor. A VisualAge view that enables a developer to implement part behavior by writing Smalltalk scripts (methods). The Script Editor shows all the objects that can be referenced by a method and all previously defined methods of a part′s class.

selector. The component of a message that specifies the requested operation. There are three kinds of selectors: binary, keyword, and unary.

sender. An object that sends a message to another object. On the level of code implementation, the sender is considered to be the sending method within the class or instance that issues the message. Contrast with *receiver*.

server. (1) A computer that provides services to multiple users or workstations in a network; for example, a file server, print server, or mail server. (2) An object that performs one or more tasks on behalf of a client. The server can be a computer (a file server), a specific process on a server, or a distributed object. A single server machine could have several different server software packages running on it, thus providing many different servers to clients on the network. See also client, network.

service. (1) A specific behavior that an object is responsible for exhibiting. (2) In network architecture, the capabilities that a layer and the layers closer to the physical media provide to the layers closer to the end user. (3) A set of service primitives that a layer provides to the layer above it.

service object. A nonvisual part that gives access to the outside world of a VisualAge for Smalltalk image.

session. (1) A connection between two application programs that allows them to communicate. (2) In SNA, a logical connection between two network addressable units that can be activated, tailored to provide various protocols, and deactivated as requested. (3) The data transport connection resulting from a call or link between two devices. (4) The period of time during which a user of a node can communicate with an interactive system, usually the elapsed time between logon and logoff. (5) In network architecture, an association of facilities necessary for establishing, maintaining, and releasing connections for communication between stations. (6) A series of CGI queries that come from the same client and belong to the same logical sequence. A session is identified by a unique session key, which is generated by VisualAge. A session begins when a client initially connects (without a session key) and ends when a specified timeout period has elapsed since the last connection.

session key. A unique string, generated automatically by VisualAge, that identifies a session. All requests and replies carry the session key in hidden HTML data fields.

session layer. The layer that provides the services that organize and synchronize

communications between functional units in different open systems located in the presentation layer.

settings view. A view of a part that provides a way to display and set the attributes and options associated with the part.

simple dialog. A dialog that takes one data record from the initiating object and sends it to a remote program, which returns the result to the initiating object. Compare to *multireceive dialog*.

Smalltalk (ST). (1) A complete programming environment for developing object-oriented applications. Smalltalk is a pure implementation of object-oriented concepts; every entity in the environment is an object. (2) The name of the programming language that the Smalltalk programming environment supports. (3) In the IBM Smalltalk programming environment, the name of the System Dictionary containing the global variables.

SNA. See *Systems Network Architecture*.

socket. (1) In the TCP/IP environment, a socket is an endpoint for communication between processes or applications. A socket that can send data to and receive data from a remote node is a connected socket. (2) Synonym for *port*.

socks. Software to intercept and redirect all TCP/IP requests at the firewall. It handles data to and from applications such as Telnet, FTP, Mosaic, and Gopher. Provides users in a secured network access to resources outside the network by directing data through the firewall. Firewall users must use client programs specifically designed to work with the *sockd* server.

software. (1) Programs, procedures, rules, and any associated documentation pertaining to the operation of a system. (2) Contrast with *hardware*.

SOM. See *system object model*.

source code. Compiler or assembler input, written in a source language. Contrast with *object code*.

source file. A file of programming code that is not compiled into machine language. A source file can be created by the specification of FILETYPE(*SRC) on the Create command. A source file can contain source statements for such items as high-level language programs and data description specifications (DDS).

SQL. Structured Query Language. A language used to access relational databases.

SQL Editor. An interactive tool for creating structured query language (SQL) statements. It consists of a set of dialogs that prompt the user for information about database tables and use that information to generate SQL statements.

statement. A language syntactic unit consisting of an operator, or other statement identifier, followed by one or more operands.

stored procedure. A procedure stored in a database system that contains SQL and other control statements.

structured query language (SQL). A language used to access relational databases.

subapplication. An application contained by another application. Using subapplications, one can organize the classes of an application into a tree of subapplications or isolate the parts of an application that are platform-specific.

subclass. A class that inherits behaviors and specifications (in other words, methods and variables) from another class. Contrast with *superclass*.

subclass type. In VisualAge or IBM Smalltalk, an indication of how a subclass inherits instance variables from its superclass.

submit button. A type of push button that can appear on a form. A submit button initiates a connection to the HTTP server and sends a CGI

query, using the data from the input fields as parameters.

subpart. A part that is embedded within a composite part.

subsystem. A secondary or subordinate system, or programming support, usually capable of operating independently of or asynchronously with a controlling system.

superclass. A class from which another class inherits behaviors and specifications (in other words, methods and variables). Contrast with *subclass*.

symbol. In Smalltalk, an object that represents a string used as a name within the system. A symbol literal is a sequence of characters preceded by the pound sign (#) with no embedded blanks, such as #George or #messageSelector.

synchronous. (1) Pertaining to two or more processes that depend on the occurrences of a specific event such as common timing signal. (2) Occurring with a regular or predictable timing relationship.

system. In data processing, a collection of people, machines, and methods organized to accomplish a set of specific functions.

system component. A component that manages storage of code and access to that stored code. A library file is a system component that stores and manages code. A user object is a system component that represents a person who can use a library.

system object model (SOM). An object-structured protocol that enables applications to access and use objects and object definitions, regardless of the programming language created that them, with no need to recompile the application.

systems management. The process of monitoring, coordinating, and controlling resources within open systems.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks.

T

TCP/IP. (Transmission Control Protocol/Internet Protocol) The basic programming foundation that carries computer messages around the globe via the Internet. The suite of protocols that defines the Internet. Originally designed for the UNIX operating system, TCP/IP software is now available for every major kind of computer operating system. To be truly on the Internet, your computer must have TCP/IP software.

team programming. Development of a system, program, or application suite by a team of two or more programmers or application developers.

tear-off attribute. An attribute that an application developer has exposed to work with as though it were a standalone part.

Telnet. An Internet protocol that lets you connect your PC as a remote workstation to a host computer anywhere in the world and to use that computer as if you were logged on locally. You often have the ability to use all of the software and capability on the host computer, even if it's a huge mainframe.

temporary variable. A variable whose scope is limited to the Smalltalk method or block in which it is defined. A temporary variable takes an assigned value.

terminal. A device that is capable of sending and receiving information over a link; it is usually equipped with a keyboard and some kind of display, such as a screen or a printer.

token ring. A network with a ring topology that passes tokens from one attaching device (node) to another. A node that is ready to send can capture a token and insert data for transmission.

tool bar. In the VisualAge Composition Editor, the strip of icons along the top of the free-form surface. The tool bar contains tools to help construct composite parts. These tools are also available through the Tools pull-down menu of the Composition Editor window.

transaction. (1) In client/server transaction processing, a business activity or set of activities that transforms a database from one state to another (for example, making an airline reservation or custom ordering an automobile). (2) In an SNA network, an exchange between two programs that usually involves a specific set of initial input data that causes the execution of a specific task or job. Examples of transactions include the entry of a customer's deposit that results in the updating of the customer's balance, and the transfer of a message to one or more destination points.

transaction program. A program that processes transactions in or through a logical unit (LU) type 6.2 in an SNA network. Application transaction programs are end users in an SNA network; they process transactions for service transaction programs and for other end users. Service transaction programs are IBM-supplied programs that typically provide utility services to application transaction programs.

Transcript window. The main controlling window in Smalltalk.

Transmission Control Protocol/Internet Protocol (TCP/IP). A set of protocols that allow cooperating computers to share resources across a heterogeneous network.

U

uniform resource locator (URL). A standard identifier for a resource on the World Wide Web, used by a Web browser to initiate a connection. The URL includes the communications protocol to use, the name of the server, and path information identifying the object to be retrieved on the server.

usability. The quality of a system, program, or device that enables users to easily understand and conveniently use it.

user interface (UI). The hardware, software, or both that enables a user to interact with a computer. In VisualAge, user interface normally refers to the visual presentation with which a user interacts and its underlying software.

user profile. A file that contains the user′s password, the list of special authorities assigned to a user, and the objects the user owns. It is used by the system to verify the user′s authorization to read or use objects, such as files or devices, or to run the jobs on the system. Each user profile must have a unique name.

user space. In OS/400 application programming interfaces, an object consisting of a collection of bytes that can be used for storing any user-defined information. The system-recognized identifier for the object type is *USRSPC.

V

variable. (1) A storage place within an object for a data element. The data element is an object, such as a number or date, stored as an attribute of the containing object. (2) In VisualAge, a part that receives an identity at run time. A variable by itself contains no data or program logic; it must be connected in such a way that it receives runtime identity from a part elsewhere in the application.

version. In the VisualAge or IBM Smalltalk team programming environment, an edition of a software component that cannot be changed. Each version has a version name. Contrast with *edition*.

view. A composite visual part. A view can display and change the underlying nonvisual objects of an application. In VisualAge, views are both the end result of developing an application and the basic unit of composition of user interfaces. Compare to *visual part*.

Virtual Storage Access Method (VSAM). An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

visible class. A class that another class can subclass or refer to by name in a method. *Visible* refers to the scope in which the class name can be used. For a class in a given application, visible classes include:

- All classes defined in the same application
- All public classes defined in any subapplication
- All prerequisite classes, including prerequisites of prerequisites to the lowest level

visual part. A part that has a visual representation at run time. Visual parts, such as windows, push buttons, and entry fields, make up the user interface of an application. Compare to *view*. Contrast with *nonvisual part*.

visual programming tool. A tool, such as VisualAge, that provides a means for specifying programs graphically. Application programmers write applications by manipulating graphical representations of components.

W

WAN. (Wide Area Network)— Any internet or network that covers an area larger than a single building or campus. See also Internet, LAN, network.

wide area network (WAN). A data communications network designed to serve an area of hundreds or thousands of miles—for example, public and private packet-switching networks, and national telephone networks.

widget. An object that provides a user-interface abstraction; for example, a scrollbar widget. Widgets support obtaining input from the user and displaying output to the user.

window. (1) A rectangular area of the screen with visible boundaries in which information is displayed. Windows can overlap on the screen, giving the appearance of one window being on top of another. (2) In the VisualAge Composition Editor, a part that can be used as a container for other visual parts, such as push buttons.

workstation. (1) An I/O device that allows either transmission of data or the reception of data (or both) from a host system, as needed to perform a job; for example, a display station or printer. (2) A configuration of I/O equipment at which an operator works. (3) A terminal or microcomputer, usually one connected to a mainframe or network, at which a user can perform tasks.

World Wide Web. (WWW) (W3) (the Web) An Internet client-server distributed information and retrieval system based upon HTTP that transfers hypertext documents across a varied array of computer systems. The Web was created by the CERN High-Energy Physics Laboratories in Geneva, Switzerland in 1991. CERN boosted the Web into international prominence on the Internet.

List of Abbreviations

AIX	Advanced Interactive eXecutive
APAR	authorized program analysis report
APPC	advanced program-to-program communication
APPN	advanced peer-to-peer networking
BMP	bitmap
CAE	Client Application Enabler
CDROM	compact disk read only memory
CG	common graphics
CGI	Common Gateway Interface
CICS	customer information control system
CLI	call level interface
CORBA	Common Object Request Broker Architecture
CPI-C	common programming interface for communications
CPU	central processing unit
CRC	class responsibility collaborators
CUA	Common User Access
DAP	developer assistance program
DBA	database administrator
DBF	database file
DBMS	database management system
DB2	Database 2
DDCS	distributed database connection services
DDL	database definition language
DDM	Distributed Data Management
DDS	data description specifications
DLL	dynamic link library
DSN	data set name
DSOM	distributed system object model
ECS	electronic customer support
EPI	external presentation interface
ESA	enterprise systems architecture
FTP	File Transfer Protocol
GIF	graphic interchange format
GPF	general protection fault
GUI	graphical user interface
HLLAPI	high level language application program interface
HPFS	high performance file system
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPD	Hypertext Transfer Protocol daemon
IBM	International Business Machines Corporation
IDL	interface definition language

ILE	integrated language environment
IMS	information management system
IPC	inter-processor communication
IPMD	IBM presentation manager debugger
ISO	International Organization for Standardization
ISP	Internet service provider
ISV	independent software vendor
ITSO	International Technical Support Organization
JDBC	Java Database Connectivity
LAN	local area network
LF	logical file
MB	megabyte
MIME	Multipurpose Internet Mail Extensions
MLE	multiline edit
MQ	message queueing
MRI	machine readable information
MVS	multiple virtual storage
NLS	national language support
NT	new technology
ODBC	Open Database Connectivity
OIDL	object interface definition language
OLTP	on-line transaction processing
OMT	object modeling technique
OOP	object-oriented programming
OOSE	object-oriented software engineering
ORB	object request broker
OS	operating system
OS/2	Operating System/2
PCS	PC/Support
PM	presentation manager
PTF	program temporary fix
RAM	random access memory
RDBMS	relational database management system
RFT	request for technology
RPC	remote procedure call
RPG	report program generator
SGML	standard generalized mark-up language
SHTTP	Secure Hypertext Transfer Protocol
SOM	system object model
SQL	structured query language
SSL	secure sockets layer
TCP/IP	Transmission Control Protocol/Internet Protocol
URL	uniform resource locator
VM	virtual machine
VMT	visual modeling technique
WWW	world wide web

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

ITSO Redbook Evaluation

VisualAge for Smalltalk Handbook Volume 2: Features
SG24-2219-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

Printed in U.S.A.