IUA CONNECTIONS

The CA-IDMS Database and Applications User Association

http://iuassn.org

Winter 2005, Number 66

FOLLOW-UP TO INTERVIEW WITH ANTHONY GAUGHAN, SENIOR V.P. AT CA

In the summer we published an interview with Tony Gaughan, the (then) new head of the Computer Associates mainframe software tools (including IDMS). One of his stated objectives was to review the policy of selling users a separate license for the SQL Option as opposed to bundling it into the IDMS package. Tony has completed this review and has come back to us with his conclusions by way of Judy Kruntorad, the IDMS Product Owner who has provided us with this response:

"The IUA Board recently followed up with Tony Gaughan to get an update regarding bundling the Advantage CA-IDMS SQL Option with the base product.

"Tony advised that after the IUA interview, he launched an initiative within CA to review the existing contracts for Advantage CA-IDMS and the SQL Option and to review the value that the SQL Option adds to the base Advantage CA-IDMS / DB Database product.

"After a long analysis and discussion at several levels of management, the decision reached was that SQL for application and end user access will continue to be an option to the base product.

"Tony commented: "The SQL Option provides significant business value to the IDMS customers who exploit it. It provides the foundation for creating new applications and for accessing IDMS data and business logic from the web. In looking at the contracts, it became apparent that the majority of our IDMS customers are already licensed for the SQL Option. Additionally, SQL technology for database administration purposes is already bundled for use with the Advantage CA-IDMS Visual DBA tool. This tool has been available at no additional charge to all Advantage CA-IDMS / DB customers since 2003.

(continued on page 2)

INSIDE THIS ISSUE

MESSAGE FROM INTERNATIONAL CHAIR

By Bob Wiklund, Tiberon Technologies

I'm very proud to have been elected to serve as the 2005 chair of the IUA. As in other years, this year is proving to be a great challenge; I'm very fortunate to have a dedicated team of IDMSr's on the board to help me.

Since Computer Associates has postponed CAWorld until this fall, the usual time for the IUA Workshop, we're working to determine where we can fit the EdOp this year. We have a few irons in the fire and will let you know when we know more.

The 2004 EdOp in Montreal was a great success. I would like to thank the IUA board who worked to put it together, Computer Associates' event planning staff, our speakers and sponsors (Computer Associates, Neon Systems, ObjEx Inc. and ISP / TACT / Cogito / VegaSoft) for a great event.

We are trying a new tack with Connections this year. Instead of publishing 3 or 4 general issues, we plan on publishing 5 or 6 issues that will be topical and a help to you. We are looking for IDMSr's to contribute articles which are worth a free year's membership. Please see the article on the upcoming issues and their topics on page 16.

I'd like to thank Laura Rochon, our out-going chair, for the great job she has done for the past 2 years. Finally, I'd like to thank those board members and volunteers who have left us this year: Kay Sussmann, Steve Nason and Jim Rice. Thanks for all your years of hard work on the behalf of the IUA.

Interview with Anthony Gaughan cont'd from page 1

"We are now working with our new Mainframe sales organization to ensure that all IDMS customers who want to derive the benefit of the SQL Option will be able to do so. We are embarking upon an extensive outreach program to contact IDMS customers that do not have the SQL Option to discuss their specific needs and to ensure that they have the right solution. CA is committed to the ongoing development and support of IDMS as well as the success of our customers.

"Tony encourages IDMS customers to contact their local sales representative if they would like to learn more about the SQL Option and the benefits that it can provide to their existing IDMS environment."

[Tony Gaughan, SVP and Business Unit Executive for Databases and Application Management at Computer Associates, was interviewed by the IUA last year. This interview was published in the Summer 2004 Connections newsletter. The complete newsletter with the interview can be viewed from the Archive page on the IUA website.]

CHECK OUT THE IUA ARCHIVE LIBRARY OF IDMS PRESENTATIONS

EDOP2004 IN MONTREAL

by Dan Hall

As the out-going chair of the Education Commission, I would like to thank everyone who helped make the EdOp2004 in Montreal a great success. There were sessions concerning the future of IDMS, what's new in release 16.0, sessions for DBA's and sessions for programmers. All the presenters of these sessions deserve a special thank you for going the extra distance. I would also like to thank all those people behind the scenes that worked very hard to make EdOp2004 a success.

I talked with a lot of people who were attending the conference and everyone seemed to enjoy the sessions. Those people from out of town seemed to enjoy Montreal.

If you attended the conference, did you fill out a survey? We need your comments! What did you like? What didn't you like? Is there something you would like to see added to the conference? Whether you were there or not, if you have a suggestion go to the IUA website (www.iuassn.org) and send us a comment. You can do this by clicking on the "About the IUA" tab and then on the "Contact Us" tab.

WWW.IUASSN.ORG

YOUR PORTAL TO IUA SERVICES AND IDMS CONTACTS

CA ANNOUNCES CA WORLD DATES

The IUA has received the following notice from Computer Associates:

"We are pleased to announce our plans for CA World^{SM} 2005. Our flagship customer event will be held at the Venetian Las Vegas and Sands Expo and Convention Center, in Las Vegas, Nevada, from November 13-17, 2005. We will be issuing a press release shortly but wanted you to have the information before it is officially announced.

"As the largest annual global gathering of CA customers, user group members, business partners, CA employees and IT leaders, CA World 2005 will provide unparalleled networking and relationship building opportunities, as well as allow attendees to explore the full-range of CA's world-class management software solutions.

"Through content-rich sessions covering CA solutions, key strategies, product demonstrations and hands-on labs, attendees at CA World 2005 will benefit from in-depth training they can implement to grow and improve their business. This year's conference will also reflect trends and issues in our industry and we intend to better leverage the value of our partnerships for the benefit of our customers. They will have an opportunity to interact with Alliance, Service and Technology Partners, as well as with CA product development, support specialists and key executives.

"The schedule and details for the conference are still being defined. We plan to have more information posted to the CA World website within the next month, including registration information and forms. Please continue to submit your ideas and input by visiting ca.com/caworld."

We will pass on more information as we receive it.

WINTER CORPORATION HAS LAUNCHED THEIR 2005 TOPTEN™ PROGRAM

You know your Advantage CA-IDMS database is large; you know it processes a huge number of transactions each day and that it can handle more concurrent users than you can count; but is your database one of the world's largest or busiest?

The Winter Corporation 2005 TopTen™ Program identifies the largest and most heavily used databases in the world. The program explores the boundaries of database scalability and celebrates the remarkable achievements of those who have built or are managing or using these databases.

The Advantage CA-IDMS development and support team encourages you to participate in the Winter Corporation's 2005 TopTen™ Program to find out how your system ranks with the world's largest databases. In past years, Advantage CA-IDMS customers have placed in the TopTen™ see results at this link.

Participation in the program is easy. If your database contains over one terabyte of user data, fill out the short questionnaire at this link and provide the validation for your responses:

http://www.wintercorp.com/VLDB/2005_TopTen_Survey/2005_TTSurvey_Frame.htm

(Find the required validation information on the Winter Corp. website. For validation of users with Advantage CA-IDMS databases, run the PRINT SPACE utility, run a supplied Advantage™ CA-Culprit™ program, and perform DCMT command screen prints.)

All participants receive a complimentary gift from Winter Corp. TopTen™ winners earn awards ranging from gourmet chocolate to customized plaques. In the last campaign, 193 awards were distributed to 63 different organizations. Participants also receive the Members Report free of charge, which is a compilation of survey findings and database technology insights.

Complete your survey with the required validation information and submit it by July 15, 2005, to qualify for being a TopTen™ winner. Plan now to complete your entry by this date.

INTEGRATION ARCHITECTURES
A Primer for the Issues Topics

by Brock Shaw

When people refer to Application Integration in relation to the web, client-server or multi-tier environments, they really mean one of a set of alternative strategies for getting applications to communicate from platform to platform. The nature of each of these strategies and their appropriateness for particular application areas must be understood before choosing a particular solution for any particular problem.

As a starting point, the strategies can be categorised in the following short descriptions:

SQL-based

The two common examples of this are ODBC and JDBC. These are essentially SQL calls wrapped up in a standard message format which most SQL-capable databases support.

EDI

This method covers all situations where a data interface has been defined to connect two different application platforms, which permits sending of

continued on page 4)

Integration Architectures continued from page 3

data files, individually or batched. The means of transmission can be by such standard means as FTP or a bespoke software solution.

Program to Program

It can be any mechanism that allows two programs to carry on a "conversation" for an application purpose. There is normally at least a pair of bespoke programs designed to carry on a particular conversation and it is in general associated with a access protocol such as TCP/IP.

Message Queue

Clearly it refers to implementations of the IBM designed MQ protocol linking two platforms, though the maker of the software on any platform is not necessarily IBM. Application programs simply send or receive defined messages to or from particular named queues and the software is responsible for all aspects of delivery of messages to the right recipient.

Having defined and summaried the methods, each is dealt with in greater detail.

SQL-based:

Such solutions are ideal for integration of applications with logical information structures already using SQL as the language. Code may for instance run on a PC or midrange platform, and need never be aware of the identity of the DBMS target of the SQL or the platform where the data resides. This aspect is handled in a separate data source definition.

The calling program requests an ODBC/JDBC service, passing the text of an SQL request. This is wrapped up in a standard message format, which can then be directed to any connected and target platform with the database. The calling program can be written in a standard programming language, e.g. C#.NET, PERL, JAVA, but can also be a tool such as Excel or Access. The return is either an error code or a set of one or more rows which satisfy the SQL request.

For IDMS users, the clear advantage is that such requests can be made to any IDMS database, including network. Hence, application integration can be very simple through IDMS Server with no programming required. If the database is a network, then an SQL schema must be defined for the existing non-SQL schema and of course there are some network facilities which may not be fully supported in SQL. It may limit some SQL requests.

Far more important in the long run may be the Table Procedure and the SQL Procedure facilities of IDMS which mean that SQL requests can be defined to be handled by application programs running under IDMS. This permits a way around the incompatible network structures but more importantly it permits the definition of SQL invoked "services", i.e. the application wishes to have some general database service performed against the IDMS database. A program is written (or existing business code used) to implement this "service" and it may be programmed in SQL or network DML language. Once developed, any application anywhere may invoke the "service" using SQL syntax.

This is ideal for situations where application programs can phrase their integration need in an SQL format and expects a synchronous response.

EDI:

This approach may be considered old fashioned but still has some appeal because of its inherent simplicity. It also permits existing defined transaction to be performed locally based on data coming from another platform, with minimum impact on the existing code. The only requirement is that both platforms have access to a data or file transfer protocol such as TCP/IP or even SNA, and software be written or acquired to transmit files between the platforms. FTP is an obvious example of a protocol widely supported. The approach however does not provide any security, error handling or recovery, and hence there are design issues for this method.

It is particularly useful for passing over batches of transactions which are in the format of an existing transaction or can be readily converted to it. All recovery and integrity issues are the responsibility of the applications or the operations. Processing schedules can however be made quite flexible on both sides.

Program to Program:

There have been facilities for communication between platforms form earliest times and these are all essentially program to program conversations. However, most rely on system software provided by vendors performing them, though there usually exist facilities which support one application program talking to another. TCP/IP is one which springs to mind but LU6.2 support provides an SNA alternative (if dated).

Under this class of integration, a program must be written on each end of a communications line: one to listen and one to transmit. If two-way communication is required, then a listener and a sender is required on both ends, and each new type of conversation needs its own support. Typically, the receiving end sets a "listener" to wait for a message and this program does not try to read a message until it is notified that one is ready to be received. It needs the sender to initiate the process by initiating connection and sending the message.

Most likely a program will perform a fairly standard sequence of coordinated steps as shown in the figure.

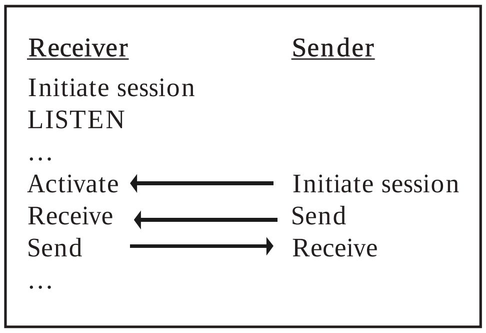
The two will synchronise when a conversation is started. It also common to allow for the use of pseudo-

(continued on page 5)

Integration Architectures continued from page 4

conversations to be established between any of the "reveive" and "send" operations to improve efficiency.

This method is ideal for any situation where integration requires a negotiation between the two applications being integrated.



Message Queue:

This approach is particularly used in order to have related applications on two or more platforms synchronise or co-operate with each other. An application program on one platform simply sends a message to a named queue. It is the responsibility of the supporting software (MQ Series, MSMQ, etc.) to handle the routing of the message, line connection, error and recovery issues and to ensure the safe delivery of the message to the other end. The reader is invited to see the related article by Laura Rochon giving her experience with MQ.

This method is ideal where one application internally decides that information must be sent to another platform, possibly many. Once sent, the application gives no further concern to the deliver of the message unless a "return receipt" message required from the receiver, which of course would need application code to manage the situation.

Conclusion

Somewhere in this list is a method which is the most appropriate to any particular integration requirement and the first priority of the designer is to make a decision as to which is the best in this case. It is hoped that the notes above may help in understanding this choice.

CONTRIBUTED
SOFTWARE
LIBRARY

Save time and use the experience of others to resolve problems.

MQSERIES – WHAT'S IT ALL ABOUT?

by Laura Rochon

MQSeries has been a buzz word for quite a while now. But, if you haven't had a chance to work with it, you might not know that much about it. Here's your chance. MQSeries products enable programs to communicate with one another, regardless of where they are located, using a consistent application programming interface. For example, in my shop, we're using MQSeries to pass data between a CA-IDMS application and an Oracle application on a HP-Unix box, and also to pass data between another CA-IDMS application and a DB2 application running on NT. This communication is done with message queuing. The exchange of messages between the applications is time independent. This means that one application can send the message, even though the other application is not running. The message is not lost. MQSeries guarantees that the message will be delivered when the other application comes back online. And that is the strength behind MQSeries.

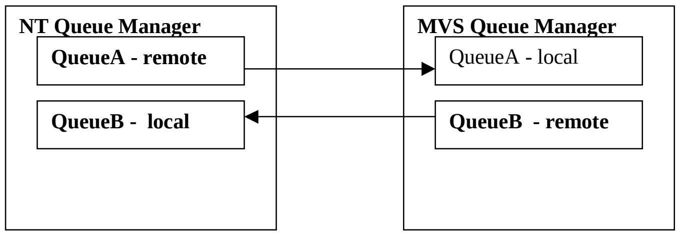
Basically, with MQSeries, we are dealing with Queue Managers, Messages and Queues. The queue manager is the part of the MQSeries product that provides the messaging and queuing services to application programs, through the MQI (Message Queue Interface) program calls. In other words, the application program talks to the queue manager for the machine that the application is running on. For example, my DB2 application on the NT box talks to the NT Queue Manager. When the DB2 application sends a message to my CA-IDMS application, the NT Queue Manager sends the message to the MVS Queue Manager where my CA-IDMS application is running. These messages are actually sent through *queues*. The queue belongs to a queue manager, which is responsible for maintaining it. A queue can be a *local* queue, meaning that it actually resides on the machine that you are running on, or it can be a *remote* queue, meaning that the queue belongs to another queue manager. This means that the queue manager must send the message on the remote queue to the queue manager responsible for that queue. For example, when my DB2 application sends a message to my CA-IDMS application, the message is being sent to a queue defined as remote to the NT Queue Manager. Therefore, the NT Queue Manager communicates with the MVS Queue Manager responsible for that remote queue. On the MVS box that queue is defined as local. Have I confused you yet?

As mentionned before, application programs communicate with the Queue Manager with MQI program calls. Basically, the application programs connects to a Queue Manager with the MQCONN command and opens a queue with the MQOPEN command. It can send messages to the queue with the MQPUT command, and retrieve messages off the queue

(continued on page 6)

MQ Series continued from page 6

with the MQGET command. And of course, there is the command MQCOMT to commit all changes sent to MQSeries. Let's use a diagram to better illustrate our example:



So in our above example, the DB2 application would issue a MQCONN to the NT Queue Manager to start a conversation with MQSeries (gees, that's sounds like a BIND RUNUNIT to me), along with a MQOPEN QUEUEA (and that could correspond to a READY to say what you're accessing). Then to send a message to the CA-IDMS application, it would do a MQPUT to QueueA. The NT Queue Manager receives the messages for QueueA, and it knows that it's a remote queue. It sends the message across to the MVS Queue Manager that stores it within the local QueueA queue. At some point, the CA-IDMS application connects to the MVS Queue Manager with a MQCONN, and opens the QueueA with a MQOPEN. It issues a MQGET for QueueA, and the MVS Queue Manager returns the message that was sent by the DB2 application. To send information to the DB2 application, a remote queue must be defined to the MVS Queue Manager, with a corresponding local queue defined to the NT Queue Manager.

There is one other concept that is important to MQSeries, it's *triggers*. You can define trigger events to MQSeries, thru the use of initiation queues. When a specified condition on a queue is satisfied, a message is sent the to initiation queue. A trigger monitor program reads the initiation queue and starts the appropriate application to process the message.

From a CA-IDMS point of view, it all depends on whether it's a batch program or an online program. However, in either case, the program needs to be put through the MQSeries pre-compiler, as well as the CA-IDMS pre-compiler. If it's a batch CA-IDMS program, the program can issue MQ commands without any problem, and you can access a Queue Manager on the mainframe. With R16 of CA-IDMS, even two-phases commit is available with the z/OS or OS/390 RRS services. For an online program running within CV, another piece of software is needed to talk to the Queue Manager. There are two products available on the market: OCA-MQSeries from Neon Systems and VEGA MQSeries from Vegasoft. Basically, both products attach an opsys subtask within the IDMS region. When a MQSeries command is executed, the command is passed to the subtask who talks to the Queue Manager. Both products allow triggers to be defined within the CA-IDMS region. Both products allow Cobol, Assembler and CA-ADS programs to call the standard MQSeries commands. And both products support Trigger Monitors within CA-IDMS.

Overall, MQSeries is a very powerfull and robust tool that allows applications to pass data back and forth regardless of their location, using a common interface. It can easily be used by CA-IDMS applications to pass data to other applications running on other platforms.

USING IDMS SERVER FOR WEB ACCESS TO CICS DATA

by Dan Hall

Sometimes we can use tools in different ways than they were originally intended. We had that happen here. We had a requirement for web access to an application running in CICS. This application allows customers to see their expedited orders. A copy of the current order database on a server, updated hourly, was not considered timely enough. We needed real time access to the data that is on the mainframe. The CICS region did not have the web interface installed, but we were already accessing IDMS data in an intranet web application on that platform. The IDMS Server apps on the NT Server called the IDMS CV on the mainframe, allowing real time access to legacy data. Since this procedure was already in place, and the Unix programmers already knew these apps, we took a novel approach to accessing the VSAM files in CICS. We defined these VSAM files to IDMS as read only files and then told the Unix programmers to go for it.

Here are all the steps we followed to accomplish this task. I hate to admit it, but we spent more time verify this was all we had to do, then actually accomplishing our goal.

- Run the old COBOL copybooks of the VSAM records into a batch IDMSDDDL job to add the to the dictionary.
- Create a small schema that defines the CICS records and areas.

```
ADD SCHEMA CICSREAD.
ADD AREA CICS-VSAM-AREA.
ADD
RECORD NAME IS TABLES-FILE
    SHARE STRUCTURE OF RECORD TABLES-FILE VERSION 1
     RECORD ID IS 1001
     LOCATION MODE IS VSAM CALC USING ( TABLES-KEY )
         DUPLICATES ARE NOT ALLOWED
     WITHIN AREA AREA07 OFFSET 0 PERCENT FOR 100
  PERCENT
     .
```

continued on page 7)

Using IDMS Server continued from page 6

```
ADD
SET NAME IS TABLES-SET
    ORDER IS SORTED
    MODE IS VSAM INDEX
    MEMBER IS TABLES-FILE
        MANDATORY AUTOMATIC
        KEY IS (
            TABLES-KEY ASCENDING )
            DUPLICATES ARE NOT ALLOWED
            NATURAL SEQUENCE
    .
```

- Define the new areas and files to the DMCL.

```
CREATE
SEGMENT CICSAPP
    FOR NONSQL
    PAGE GROUP 0
    ;
     CREATE
     FILE CICSAPP.FILE01
         ASSIGN TO FILE01
         DSNAME 'CICS.VSAM.FILE01'
         DISP SHR
         KSDS FOR CALC
         ;
     CREATE
     PHYSICAL AREA CICSAPP.AREA01
         PRIMARY SPACE 6000 PAGES  FROM PAGE 4000001
         MAXIMUM SPACE 6000 PAGES
         PAGE SIZE 0 CHARACTERS
         WITHIN FILE FILE01
             FROM 1 FOR ALL BLOCKS
         ;
MODIFY DMCL IDMSDMCL.
       INCLUDE SEGMENT CICSAPP
           DEFAULT BUFFER CICSAPP-BUFFER
           ON STARTUP SET STATUS TO RETRIEVAL
           ON WARMSTART MAINTAIN CURRENT STATUS
           DATA SHARING NO
           DEFAULT SHARED CACHE NULL
    .
```

Once the DMCL was generated and then new copied into the CV, we had web access to the CICS VSAM data. There was no outage to either CICS or IDMS to implement this. The cost to allow web access to CICS VSAM files was 1 or 2 days work from the IDMS DBA. As mentioned before, most of this time was reading manuals to make sure we were not missing any steps. Since we were already using IDMS Server, nothing else needed to be added or changed to allow the Unix Servers to have SQL access to CICS VSAM data. Our internal customers were very happy with the speed at which we were able to "web enable" the CICS data. They were under the misconception that any changes on the mainframe took weeks, if not months, to perform. Also, they certainly enjoyed the low cost associated with the web access to their legacy mainframe data.

AN OVERVIEW OF TCP/IP

by Terry Schwartz

TCP / IP (Transmission Control Protocol / Internet Protocol) is the basic communication language of the web. When you utilize TCP / IP, the data you are sending is assembled into a packet in which is embedded the address of the destination of the packet. For example, when you bring up a web site in a browser, you are exchanging TCP / IP packets back and forth with the IP address that represents that web site (i.e. 192.168.33.25) TCP / IP packets are simply buffers of data that must be dissected by a program on the receiving end. Programs communicate over TCP / IP by using simple commands such as Get Socket, Connect, read, write, and close socket.

In OS / 390 there is one TCP / IP address space that receives and sends all incoming / outgoing TCP / IP packets. The TCP / IP address space determines where to send a packet of data by the port number that is included in the message. For example if you code a browser URL like HTTP: / / 192.168.33.25:3000 , the number 3000 after the colon tells TCP / IP to route the packet to the program listening on port 3000. On the mainframe when systems like CICS and IDMS start they inform TCP / IP of the port(s) that they will be using to communicate.

Computer Associates added a TCP / IP interface to CA-IDMS in version 16. This allows ADS / O, COBOL, and Assembler programs to communicate with any platform using the TCP / IP protocol. To facilitate TCP / IP communications in CA-IDMS a line with a type of SOCKET is added to the sysgen. The line definition contains the port number that tells TCP / IP to route messages to that port to IDMS. Enough LTERM / PTERM pairs are added to the line to support the maximum number of concurrent TCP / IP sessions. The line definition also contains the name of the user task that IDMS will pass any incoming TCP / IP packets. There are two types of programs that can utilize the TCP / IP interface. Listener programs that receive incoming packets and client programs that send data out to a host.

The new TCP / IP interface opens IDMS up to new functionality. Below are a few of the uses that the TCP / IP interface can be used for:

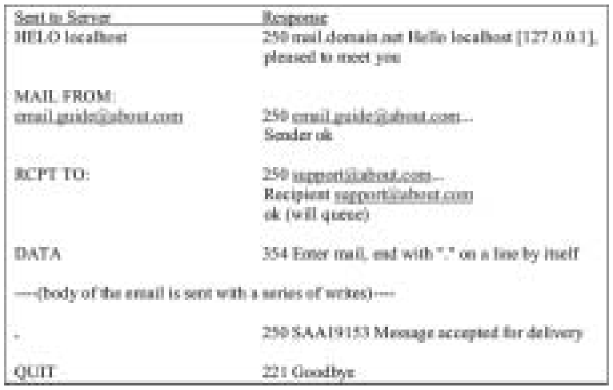
- Send and receive data from / to a Java or Web application
- Real time Electronic Data Interchange
- Data exchange with programs on Unix or AS400
- Send and receive Email

One functionality that I have implemented in our shop is sending email from IDMS. We use this as a simple work flow tool to notify users that they have items to approve. Email is sent using TCP / IP and SMTP (Simple Mail Transfer Protocol). SMTP is a structure that governs the

(continued on page 8)

An Overview of TCP/IP continued from page 6

conversation between your program and a mail server. TCP / IP is used to connect to the mail server and then a conversation with the mail server is held using TCP / IP packets. Packets sent to the mail server contain SMTP



Sent to Server	Response
HELO localhost	250 mail.domain.net Hello localhost [127.0.0.1], pleased to meet you
MAIL FROM: email.guide@about.com	250 email.guide@about.com... Sender ok
RCPT TO:	250 support@about.com... Recipient support@about.com ok (will queue)
DATA	354 Enter mail, end with "." on a line by itself
----(body of the email is sent with a series of writes)----	
.	250 SAA19153 Message accepted for delivery
QUIT	221 Goodbye

keywords and a response is then read from the server with a response to your message. Example of these keywords / responses conversations are as follows:

I will be posting the code for sending email from IDMS in the IUA's User Contributed library. As you can see the new functionalities built into CA-IDMS are opening up new avenues of communication and accessibility to data kept in IDMS databases. I encourage you to find new uses for CA-IDMS so that you can bring more value to your corporate IT solutions.

HAVING A GO WITH THE RELEASE 16.0 TCP/IP LINE DRIVER

by Gary Cherlet

Most readers would be aware that one of the major new features in Release 16.0 of CA IDMS-DC is the TCP / IP line driver. This is a great addition because TCP / IP is a leveler in terms of communication between disparate types of hardware and operating systems in an "open world". The TCP / IP line driver will allow your mainframe, online applications to operate as equal partners in today's mutli-varied computing landscape.

This is not a tutorial on TCP / IP – but there are a few things that we should probably mention by way of introducing the protocol for the uninitiated. At its simplest it is a non-propietary client:server communications protocol – in contrast to APPC which is a proprietary protocol.

In a similar way that your Subschema Control area represents a database unit of work once you've done the BIND, a SOCKET represents a TCP / IP unit of work – not dissimilar to an APPC conversation.

Outbound Messages

When establishing a TCP / IP conversation you initialize the API, allocate a SOCKET and then point TCP / IP to the IP Address and Port that you wish to communicate with. For outbound messages TCP / IP will dynamically allocate you to any free port. Depending on the application you will either simply send your message to the indicated destination, or you will send it and wait for a reply – probably the more common situation in a distributed application.

Inbound Messages

Your outbound message was directed to a specific IP Address and Port – there is a "listener" on that port that will read and process incoming message traffic. If there is nobody "listening" on that Port at that IP address – an error is returned – similar to the case when you point your browser to a URL and it comes back with an error because the server is down.

The Generic Listener

The "listener" is the "server" in this scenario – it sits there waiting for a message to arrive for it to act on. You have two basic choices in the design of your listener:

1) It can both Read and Process the message – and then Write the reply back to the originator, or

2) The listener can attach another task, pass the Socket to that task, and then establish another Socket and continue listening for the next message.

It should be clear that there is the potential for performance problems in the first case, and that there is a fair deal of technical knowledge required in the second case.

This is the beauty of the "generic listener" that is provided by CA. In system generation you can specify a PTERM that will have a listener active on a user specified Port, and the name of the Task (with its associated program already defined in sysgen) that will be invoked when a message arrives on that port.

This allows you as the developer to code the "server" end of a client:server application with the minimum amount of technical knowledge and a minimal TCP / IP verb set – basically at the READ / WRITE / CLOSE level – no need for the equivalent of an "OPEN" because this was already done for you by the listener. So you can see there is a very small learning curve.

Message Formats

Once again you've got two basic choices – you can go with the tightly coupled, data dependent design using fixed format records, kind of like a sequential file with only one record in it (the inbound or outbound message). This choice of course is very inflexible and in today's dynamic world is not very desirable – since somebody who wants to use your server must know and understand your record layouts.

(continued on page 9)

Having a Go with the Release 16.0 continued from page 9

Far more desirable is to use a “data independent” format such as XML (eXtensible Markup Language). The beauty of XML is that it gives you a great deal of data independence. The incoming document can have more information in it than your application requires – but as long as it has the minimum content that you require you will be able to process the transaction.

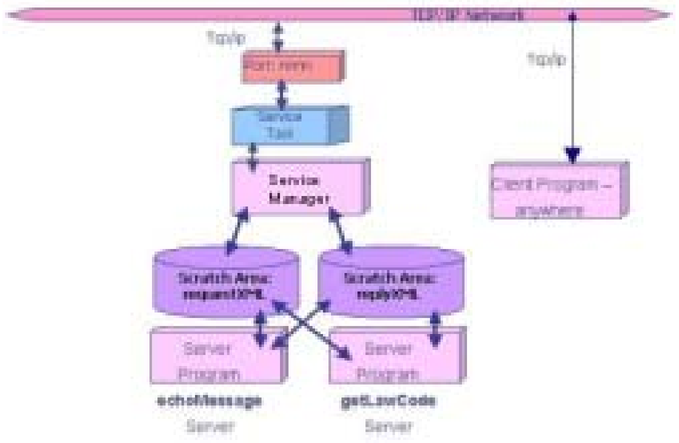
In a fixed format design if you get too much information you will have a “wrong length record” problem, and data may be shifted to the left or right of where you expect it to be. When you process an XML document from one source the data content may be in a different sequence or location within the document than the same document from another source – but this is all sorted out by the “XML Parsing” process.

Parsing XML

In an IDMS-DC application you can use the Cobol XMLPARSE verb. This is a SAX style parser – Simple Access to XML. Basically it starts at the beginning of the document and works its way through – each start and end “tag” for identifying data content is an “event” which you can choose to process or not.

This can be contrasted to the DOM processors (Document Object Model) which parse the entire document and effectively create a hieararchical database in memory that you can “navigate” from parent to child down any of the “branches” of the tree.

The SAX style parser provided by Cobol is easy enough to learn to use – certainly simple documents can be parsed and the required data content can be extracted with relatively small amounts of code. I expect that more complex documents with variable numbers of “children” in the tree may prove to be more “challenging”.



Sample Architecture

The diagram above shows one architecture that helps to simplify application coding by isolating technologies (note: all programs are Cobol). The “Service Manager” is the only program (other than the “client”) that needs to know how to speak TCP / IP. It works out the right program to invoke and passes the inbound XML document through the “requestXML” Scratch Area.

The “Server Program” is the only program that needs to understand the peculiarities of the specific XML document that carries the data content for the transaction that it will process. The results of the transaction are formatted as XML and passed back to the “Service Manager” through the “replyXML” Scratch Area.

One advantage of this architecture is that other applications running in the same IDMS-DC environment can use the Server Programs through the Scratch Area interface – so you really do only need one copy of the code to service all environments – be they 3270, green screen based, or mid-range or other mainframe application based.

Bottom Line

The Release 16.0 TCP / IP line driver allows you to code mainframe applications that can act as either a client or server in today’s highly networked world. By using XML as the “common language” at each end of the conversation these applications will be able to stand the test of time as message formats change and grow. Existing applications can continue to process as they always have without change – while new applications that use the same message can take advantage of the new data content.

With enough time, and enough space we could try to explain how all of this fits into the WWoA – Wonderful World of Acronyms: UDDI, WSDL, SOAP, SOA, MOM and so forth. But for now this hopefully gives you some idea of what you can do with the TCP / IP line driver in Release 16.0, and how you can put it to good use almost straight away.

THE DUMBEST SOLUTION EVER?

by Gary Cherlet

This is a story about IDMS system throughput issues that required us to take an approach that is contrary to the usual approaches that I have seen taken – in fact it is probably the dumbest solution to a performance problem that I have ever seen.

Police trainers at the South Australian Police Academy regularly have 30+ Officers, spread across two training rooms, logged on to the system. The number of deadlocks and the number of times that the system froze completely had become unacceptable. The problem is exacerbated by the fact that all the trainees are doing the same type of work – and so hit all the database “hot spots” very hard for such a small number of users (our production system regularly has 520-530 users on at the same time).

continued on page 10)

The Dumbest Solution Ever continued from page 9

The level of deadlocking made the job difficult, if not at times impossible for the trainers. The number of ABORTs and the amount of lost work would be very frustrating for the students, and for new Officers would not create a very good first impression of the computer systems that they will be expected to use.

Normally, all other things being equal in terms of program and storage pool sizes and buffering and IO subsystem performance for example, you start by tuning MAX TASKs so that you have the smallest number that allows you to not to sit permanently in the MAX TASK condition. This is an attempt to help the system run faster by keeping dispatch and resource chains as short as possible.

The problem with starting up lots of tasks, especially if you are running close to the limits of the amount of available CPU cycles, is that the system works harder and runs slower, the tasks take longer to complete and so have a greater chance of deadlocking with each other, which the system has to work harder to manage and resolve. So generally – less (tasks) is better.

We tried looking into application changes – but this proved to be impossible for a number of fundamental design as well as political issues. So we were left with trying to find a system level solution to the problem.

Attempt 1

Our first attempt was to try making task ADS2 non-concurrent to try to enforce some single-threading – this actually made the problem worse because the students were blasted right out of their emulator sessions when they got a message about the task being non-concurrent. Next - by trial and error, we determined that the smallest number that we can set MAX TASKs to, and not clag the system, is 22.

This did not quite force single threading to avoid deadlocks - but came pretty close (there are 19 system tasks). At that number we can not afford anybody to get into the Performance Monitoring tool (PMRM) - so set task PMRM disabled. We have a CLIST that enables PMRM, bumps MAX TASKs by 1 on entry, and reverses the procedure on exit.

We modified the control record for “Hunter/Killer” to “look around” every 60 seconds instead of 150 seconds, the “maximum time” that a task will be allowed to execute has also been lowered from 150 seconds to 60 seconds. This should help to “clean up” the system faster when long queues of users build up waiting for access to particular database records.

Attempt 2

Still looking for “quick fix” options to apply specifically to the Training environment. We tested a system generation option for the ADSOMAIN program that forces it to “single thread”. ADSOMAIN is the program that “drives” all ADS applications for all users. By forcing it to “single thread” we are telling the system that it’s OK for lots of users to start executing an ADS task (i.e. press <enter> or any other attention key), but whenever that task wants to use ADSOMAIN it can only do so if there is no other user of that program.

Our problem with tightening up the maximum number of concurrent tasks to force single threading had now reversed. We now wanted to allow a large number of tasks to start up - but force them to pass through ADSOMAIN one at a time. The appearance to the users, that is the students and trainers, is that they are all running their tasks at the same time - there is no apparent difference or change in appearance of the online application processing. We then implemented the following changes to the Training environment:

1) Change system generation to make ADSOMAIN NONCONCURRENT, and

2) Increased the maximum number of concurrent tasks from 22 to 26 (although we have since discovered that 30-40 or more was a better number).

Attempt3

I noticed that even with only 7-9 users on the system Hunter/Killer was cancelling user tasks because they were waiting for a program to be loaded. This would no doubt have been waits for ADSOMAIN (the program we are using to force a “single server queue”).

In an earlier effort to clean up database deadlocking (waiting) earlier rather than later one of the changes made was to shorten the amount of time that Hunter/Killer would allow tasks to be active from 150 seconds to 60 seconds. I have since changed the amount of time from 60 seconds to 90 seconds to allow for the waits for access to ADSOMAIN.

Also – we applied put an “Hunter/Killer override” to allow this specific condition - in other words tasks that are waiting for a program to load will be allowed to run indefinitely (although we will be sent “warning” e-mails) - but other tasks waiting for other reasons will be cancelled after 90 seconds.

Final Word?

In recent weeks Police have had a number of training sessions in the Training environment and there have been no “events” in the log: no deadlocks, no abends caused by our adjustments, no task terminations by Hunter/Killer. It looks like we are getting a handle on TRN even without the recommended application changes being made.

So our final solution was to, as much as possible, create a situation where we forced the system to all but single thread, we allowed a lot of tasks to start up to cater for

continued on page 11)

The Dumbest Solution Ever continued from page 10

the situation where many users are waiting for access to ADSOMAIN, but lots of active tasks didn't matter because in this scenario because they don't use any CPU or acquire resources that might cause deadlocks. Dumb or what? Who would ever think to make ADSOMAIN non-concurrent?

SECOND THOUGHTS

Denise read the "dumb idea" article and decided that people might need to know what "Hunter/Killer" is (several references to it) - so here is a "teaser" that might precede the "dumb idea" article. As mentioned in the last paragraph - we could make the "package" available to UCL if there's any interest in it. Don't worry if this one isn't useful - it was a 5 minute <cut> <paste> job from the documentation that comes in the "package".

HUNTER-KILLER FACILITY

by Gary Cherlet

Task GUT0042T tries to identify ("hunt") and terminate ("kill" with a *dcmt v active task terminate* command) online tasks which have been running in the system for too long.

There are other ways of trapping tasks that have been running for too long in IDMS-DC/UCF systems, but the most likely way would be with *system resource limits for online tasks*. Unfortunately there is a performance penalty of around 5-7% additional CPU for the region – in addition to another 10% (varies site to site and also depends on options in sysgen) to have statistics collection ON. Another technique would be to put code in the task termination exit to do a "look around" for online tasks that have been running for too long. This has the drawback of doing a bit too much work in this exit and may have performance consequences.

How it Works

GUT0042T is executed automatically when the CV/DC system starts up, as a *startup autotask*. Each time it runs this program CALLs an assembler program (GUT0062O – the "hunter") – this program looks through system control blocks to determine if there are any tasks that have exceeded one of the following limits: duration (wall clock time), IOs, calls to DBMS or page requests. Any such tasks are reported back to GUT0042O (the Cobol driver), which then determines whether to do one of 3 things:

- Nothing (ALLOW override)
- Report – to the log and by e-mail (WARN override)
- Terminate ("kill" the task) everything else - with appropriate notifications

Successive executions of GUT0042T are scheduled programmatically with a *set timer interval* command - where the time interval is based on a value obtained from a queue record, modifiable by a mainline ADS dialog.

Exempt Tasks and Programs - Overrides

It is not desirable to terminate some IDMS tools that run in conversational mode as they are used by developers and administrators alike to support the various environments. In development environments you want to let compilers finish, in production environments you (we) don't want them running at all. Just as an example though here are some typical development "overrides":

- **PMRM and OPER are not reported and not terminated**
- **ADS/Alive and DMLO are reported but not terminated**
- **Compilers are excluded (ADSC, ADSA, MAPC, IDD, Schema, Subschema)**
- **Specific known dialogs in production are "churners" and are "allowed"**

A second mainline dialog is available to allow authorised users to modify these "override specifications as to which tasks or programs are either "allowed" (never terminated and no warnings) or are to generate warning messages sent out by e-mail to specified production support personnel.

Hunter/Killer has previously been made available to other users through IDMS-L – if anybody is interested we can update the last distribution "package" and make it available to the IUA's User Contributed Library – any takers?

BE AN IUA VOLUNTEER

DO SOMETHING YOU LIKE TO DO AND EARN A FREE YEAR'S MEMBERSHIP

IDMS-L

WHERE IDMS TECHIES MEET

HANDLING CA-IDMS SERVER ERRORS IN ASP AND ASP.NET PROGRAMS

by Kay Rozeboom

ASP and ASP.net are just like any other language in that accessing data often generates errors. A well-written program will trap these errors for two purposes: 1) to inform your application code that an error has occurred so that it can proceed in an appropriate manner, and 2) to display or log details about the error for further debugging. This article will demonstrate how to do both. It assumes that you have some programming experience in ASP and / or ASP.net.

CA-IDMS Server often generates more than one message for a single error. The information that you need is not always in the first message. This means that your error-handling code must loop through a series of error messages. The sample code below shows how to do this in both ASP and ASP.net.

It has been my experience that you can save a lot of time by including error-handling code in your program from the very beginning, rather than trying to add it later. You will have better luck persuading programmers of this if you supply them with error-handling routines that they can call from, or copy into, their programs.

This article has two sections: the first section is for the old version of ASP, and second section is for ASP.net. In both cases, the error-handling code is encapsulated into subroutines. Each section also includes a fragment of sample application code that calls the subroutines. Please remember that these are code fragments that illustrate error-handling. They are not meant to be all-inclusive.

ASP (old version)

The sample ASP code consists of three pieces:

1) The first piece defines some working storage fields to be used for error handling. It should be placed near the beginning of your program, with your other 'dim' statements.

2) The second piece consists of two callable subprocedures that process the errors. It should be placed near the end of your program, with your other called subroutines.

3) The third piece is a sample section of application code that shows how to call the subprocedures.

Notes about the sample code:

1) If you put the first two pieces (the working storage fields, and the subprocedures) into ASP 'include' files, you can call them from multiple programs.

2) The "On Error Resume Next" statement turns off the regular ASP error-handling. So you must be sure to check for all potential errors yourself.

3) Note the "MoveFirst" command in the sample application code. Most programmers skip this step, and just use MoveNext to get every row. However, when accessing IDMS data, it is important to use MoveFirst for the first row. MoveNext contains a bug, which Microsoft has no plans to fix: If the first row of data returned contains a data exception (S0C7-type of error), MoveFirst will catch the error, but MoveNext will not.

```
'  ***********************************************************************
' *  CAServer working storage fields (piece #1) - start
'  ***********************************************************************

dim  CAServer_connection
dim  CAServer_error
dim  CAServer_did_error_occur
dim  CAServer_error_message_table(20)
dim  CAServer_error_message_sub
dim  CAServer_number_of_error_messages
dim  CAServer_display_message_sub

CAServer_did_error_occur  =  "no"

'  ***********************************************************************
' *  CAServer working storage fields (piece #1) - end
```

(continued on page 13)

```
'  ***********************************************************************


'  ***********************************************************************
' *  CAServer called code (piece #2) - start
'  ***********************************************************************

'  ***********************************************************************
' * Subprocedure "CAServerErrorSub"
' *    - Check for errors.
' *    - Loop through errors, storing messages in a table.
'  ***********************************************************************

sub  CAServerErrorSub(CAServer_connection)

   CAServer_did_error_occur = "no"
   CAServer_error_message_sub = 0
   CAServer_number_of_error_messages = 0

   If CAServer_connection.Errors.Count > 0 Then
     CAServer_did_error_occur = "yes"
     CAServer_error_message_sub = 0
     For Each CAServer_error in CAServer_connection.Errors
       CAServer_error_message_table(CAServer_error_message_sub) =
         CAServer_error.Description
       CAServer_error_message_sub = CAServer_error_message_sub + 1
     Next
     CAServer_number_of_error_messages = CAServer_error_message_sub
   End If

end sub

'  ***********************************************************************
' * Subprocedure "CAServerDisplayErrorMessages"
' *    - Loop through error message table, displaying messages.
'  ***********************************************************************

sub  CAServerDisplayErrorMessages

   response.write ("<table border=''1''>")

   If CAServer_number_of_error_messages > 0 then
     For CAServer_display_message_sub = 0 to CAServer_number_of_error_messages
       response.write ("<tr> <td> " _
         & CAServer_error_message_table(CAServer_display_message_sub) _
         & "</td> </tr>")
    Next
   Else
     response.write ("<tr> <td> No CA-Server error messages. </td> </tr>")
   End If

   response.write ("</table>")

end sub

'  ***********************************************************************
' *  CAServer called code (piece #2) - end
' ***********************************************************************
```

continued on page 14)

Handling CA-IDMS Server Errors continued from page 13

```
' ***********************************************************************
' *  Sample application code (piece #3) - start
' ***********************************************************************

  On Error Resume Next

  myConnection.Open ws_connection, ws_userID, ws_password
    Call CAServerErrorSub(myConnection)

    if CAServer_did_error_occur = “yes” then
      Call CAServerDisplayErrorMessages
    else
      myRecordset.open ws_SQL_Command, myConnection
      Call CAServerErrorSub(myConnection)

      If CAServer_did_error_occur = “yes” then
        Call CAServerDisplayErrorMessages
      else
        myRecordSet.MoveFirst
        Call CAServerErrorSub(myConnection)

        If CAServer_did_error_occur = “yes” then
          Call CAServerDisplayErrorMessages
        else
          ' – further processing here –

' ***********************************************************************
' *  Sample application code (piece #3) - end
' ***********************************************************************
```

ASP.net

The sample ASP.net code consists of two pieces:

1) The first piece is a callable function that processes the errors. It should be placed near the end of your program, with your other called subroutines.

2) The second piece is a sample section of application code that shows how to call the function.

Notes about the sample code:

1) If you put the first piece (the callable function) into a class library, and compile it separately, you can call it from multiple programs.

2) Note that we are passing the line break character to the function. This allows the function to format the CA-Server error messages for other types of processing, such as writing them to a log file or sending them in an email. This will also let you use the same function for VB.net forms such as a text box or pop-up message box.

```
'  ***********************************************************************
' *  CAServer called code (piece #1) - start
'  ***********************************************************************

     Public Function FormatError(ByVal myException As OdbcException, _
            myLineBreak As String) As String
         Dim i As Integer
         Dim messageText As String
          messageText = “”
          For i = 0 To myException.Errors.Count - 1
               messageText = messageText & “Message #”
               messageText = messageText & (i + 1)
               messageText = messageText & “: “
```

continued on page 15)

Handling CA-IDMS Server Errors continued from page 14

```
            messageText = messageText & myException.Errors(i).Message
            messageText = messageText & myLineBreak
        Next i
        Return messageText
    End Function

' ***********************************************************************
' *  CAServer called code (piece #1) - end
' ***********************************************************************


' ***********************************************************************
' *  Sample application code (piece #2) - start
' ***********************************************************************

    myLineBreak = "<br />"
    OK_to_continue = True

    Try
        objConnection.Open()
    Catch myException As OdbcException
        messageText = messageText _
          & "CA-Server connection error: " & myLineBreak _
          & FormatError(myException, myLineBreak)
        OK_to_continue = False
    Catch myGenericException As System.Exception
        messageText = messageText _
          & "CA-Server connection error (system): " _
          & myGenericException.Message & myLineBreak
        OK_to_continue = False
    End Try

    If OK_to_continue = True Then
        objDataAdapter.SelectCommand = New OdbcCommand
        objDataAdapter.SelectCommand.Connection = myConnection
        objDataAdapter.SelectCommand.CommandText = mySQL
        objDataAdapter.SelectCommand.CommandType = CommandType.Text
        Try
            objDataAdapter.SelectCommand.ExecuteNonQuery()
        Catch myException As OdbcException
            messageText = messageText _
              & "CA-Server SQL command error: " & myLineBreak _
             & FormatError(myException, myLineBreak)
            OK_to_continue = False
        Catch myGenericException As System.Exception
            messageText = messageText _
              & "CA-Server SQL command error (system): " _
              & myGenericException.Message & myLineBreak
            OK_to_continue = False
        End Try
    End If

    If OK_to_continue = True Then
        ' – further processing here –
    End If

' ***********************************************************************
' *  Sample application code (piece #2) - end
' ***********************************************************************
```

TOPICS FOR *CONNECTIONS* ISSUES

by Brock Shaw

The IUA Board has decided to change the focus of the *Connections* issues for the following year or so, to see if this meets our members needs and interests better. It has tentatively set up the following topic areas for articles in the proposed issues of *Connections* below.

The first issue in this format (this one) has been on the topic below, though not all of these subjectss have been covered in articles:

Issue Topic	Article Subject
Application Integration and the Web	•Integration Design •TCP / IP •MQ Series •ODBC / JDBC •EDI •XML •Working with Oracle • Future Directions •“Webifying” •“GUI-fication”

If members still wish to contribute on any of these topics now, it is not too late. Articles can be published in future issues along with the focus topic articles.

The planned issues are proposed as shown below, and we are now seeking articles for any of these topics and subjects:

Issue Topic	Article Subject
Application Integration and the Web	•Integration Design •TCP / IP •MQ Series •ODBC / JDBC •EDI •XML •Working with Oracle • Future Directions •“Webifying” •“GUI-fication”
Application Development	•ADS Application Design •ADS Tips •Batch Issues •Accounting Principles •SQL •Configuration Management
DBA Topics	•DBA tricks of the trade •Database Design •24x7 Processing •R16 Features (list) •R16 Experience
Safety and Security	•IDMS Security Overview •User Security •Resource Security •Dictionary Security •Backup issues •Recovery issues •Journalling issues
IDMS Internals	•Operating System Basics •IDMS Exits •CSL Highlights •Indexes and Rebuilding •Journalling
Performance and Tuning	•Performance Tips •Approach to Tuning •System Performance •Database Performance •Application Performance •Real-time Performance
3rd Party Products	[sources: vendors and users; no sales pitches]

All members are invited to contribute, and below is the schedule of issues currently planned.

Issue	Issue Topic	Call for Articles
05/1	Application Integration & the Web	10 Jan 2005
05/2	Application Development	10 Apr 2005
05/3	DBA Topics	11 Jul 2005
05/4	Safety and Security	11 Oct 2005
06/1	IDMS Internals	10 Jan 2006
06/2	Performance and Tuning	10 Apr 2006
06/3	3rd Party Products	10 Jul 2006

Remember that publication of an article is an automatic 1 year free membership of IUA, so please do respond soon to cover this year’s fee.

SOME INSIGHTS ON JOHN SWAINSON

Because of the importance of the good health of CA to IDMS and because the new CEO is a complete outsider some extracts have been made here from a recent articel in eWeek.com. The selection of the Q&A below is reproduced for members information. In order to use any of this material, you are directed to the source website.

After months of uncertainty sparked by corporate scandal, Computer Associates International Inc. earlier this month embarked on a new era, naming CEO-elect John Swainson to his official post as president and chief executive. A veteran of IBM’s software division, Swainson said he comes prepared to restore credibility and accountability at CA by paring offerings to a few core

continued on page 17)

Some Insights on John Swainson continued from page 16

technologies and remaking the culture of the Islandia, N.Y., company. Swainson sat down with eWEEK News Executive Editor Chris Gonsalves and Senior Writer Brian Fonseca before his appearance at LinuxWorld in Boston last week to discuss the job ahead.

How would you characterize your role at CA? Is it rehab, rescue, maintenance, other?

It sure as hell isn't maintenance. Maintenance implies I'm here to protect things. I don't think that's at all why the board brought me here or what people are expecting. Rescue may be overstated and overly dramatic, but it's more to the point.

CA has a troubled past. Companies don't voluntarily get themselves into this kind of dilemma where 15 of their senior executives are indicted or otherwise implicated in accounting fraud and various other nasty things, including trying to mislead the government. This is a company that has some serious problems to remedy, and, in that sense, I am coming in as part of a team that is doing that. It's not John, personally, is the savior of the world. It's me and the management team and the employees working together to fix some problems and to build a new CA.

Of late, we've seen companies such as Hewlett-Packard Co. get into trouble by being pretty good at a bunch of things and excellent at very few. How will CA, with its notoriously large portfolio, avoid that trap?

We'll concentrate on a small number of things that we can get really good at. In our case, it's two: system management and security. Today, we are the market leader in system management, and we're tied for leadership in enterprise security, so we've a got a decent starting place. We're going to build on that by acquisitions, as you saw with Netegrity [Inc.]. We're also going to build by investing in development of products in those segments. We're going to try to do those two things astonishingly well and use that to rebuild the franchise.

We have a very important install base of mainframe database customers, but I'd be quick to tell them that the mainframe database is not where we're going. We'll support them as long as they want us to, but we're not going to really fight that battle. It's yesterday's battle.

Is there a culture change going on at CA?

I think so. CA didn't have a strong culture. It's a function being built by acquisition over the course of a relatively short period. IBM had a chance over 100 years to build a strong culture. One of the things I have to do is make sure we build a culture that is customer-focused-that is about being a trusted adviser to our customers. To be an ethical company in everything we do. That all needs to be inculcated into the body of the organization.

The good news is that our employees have seen firsthand what happens when a company loses its way. It's not as though you have to go out and convince them that something is needed. It's about showing them a vision of what the company can do and their role in it.

What parts of your experience from IBM are your bringing to the task at CA?

I'm not consciously trying to bring any parts of IBM culture to CA. IBM does a lot of good things around process and focus on customer. I'll bring those. On the other hand, one of the things I've learned from CA is the resourcefulness of the people, their entrepreneurship, their ability to endure almost any kind of [obstacle] management throws at them and be able to persevere.

Copyright (c) 2005 Ziff Davis Media Inc. All Rights Reserved. See their website: http://www.eweek.com

IUA CONNECTIONS

Communications Commissioner
Linda Campbell, Informatix

Editor
Brock Shaw, Torridon Associates (UK)

Graphic Design
Rebecca Shaw, Torridon Associates (UK)

IUA International Chairperson
Bob Wiklund, Tiberon Technologies

IUA Connections is a quarterly publication of the CA-IDMS Database and Applications User Association (IUA). It is designed to promote its members' objectives. *IUA Connections* is not responsible for the opinions expressed by its writers and editors.

Information User Association
401 N. Michigan Ave.
Chicago, IL 60611-4267
Phone: 312 / 321-6827
Fax: 312 / 245-1081

Internet: iua@iuassn.org
www: http: / / iuassn.org