IDMS Connections

THE publication of the IDMS User Association

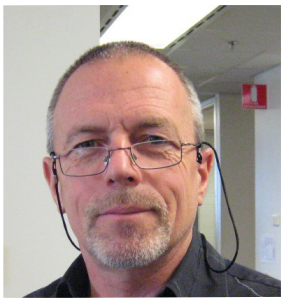
The CA-IDMS Database and Applications User Association

https://communities.ca.com

September 2011, Number 75

INSIDE THIS ISSUE

LETTER FROM THE EDITOR



As this issue "goes to press" the IUA is now represented in the cyber world in two ways:

1. https://communities.ca.com/web/ca-idms-iua-eiua-global-user-community - **the** place to go for **"everything IUA"**:
   - Current discussion threads from IDMS-L and original posts to the Message Board
   - Archives of "IUA Connections" and "IDMS Connections – including PDF's of back issues that were previously published only as "hard copy"
   - User Contributed Library
   - Up-to-the minute news
   - Wiki and Blog
   - Photos, pictures of memorabilia, etc, etc
2. mailto: idms-l@listserv.iuassn.com the long standing IDMS-L List Server – which contains current discussion threads that originate either on IMDS-L or on the IUA/EIUA Message Board

Starting up new "threads" on the Message Board is still far too rare an occurrence (maybe "NOOAK posts"?) – but I remain hopeful that people will realize that posts on IUA/EIUA Community Message Board also go to IDMS-L, and vice-a-versa, **so it doesn't matter where you start your discussion thread!**

Speaking of IDMS-L! In this issue we feature a number of threads extracted from "the archives" that I keep for Connections – when it looks like the topic is particularly relevant and may be worth re-visiting at some point in the future. So – for the threads that made it into this issue – their future has arrived. I hope you enjoy these "oldies but goodies".

(continued on page 3)

She already knows how to use IDMS!

Your bosses may think IDMS is "legacy," but do they know this graduate can write Java or .NET applications - *right now* - that use IDMS data? Not only that, she'll do it several times faster than she could using a relational database!

Using our NoDML Document Oriented technology, you open a document, update it, and save it with three calls. Even a newbie can do that! To learn how you can build world-class offerings using our Cloud Computing technology for IDMS

call us at +1 (480) 588-7776
or email us at: info@obj-ex.com

www.obj-ex.com

Photo by: BdwayDiva1 at flickr.com

ObjEx INC.

Experts in IDMS Integration

MESSAGE FROM INTERNATIONAL CHAIR

By Linda Casey

What does the IUA mean to you?

In early August, Terry Schwartz, functioning as the chair of the nomination committee, posted a request for nominations for the IUA board positions available in this year's election cycle. While we will be voting via a poll at the communities' site, we will be concluding the elections at our annual meeting CA World.

Much to my chagrin, there was little response to his request. So let me ask you, the membership of the IDMS User Association, a few questions:

1. What can we do as a community to revitalize membership involvement?
2. What do you, as members of the IUA what from the IUA?
3. What do your fellow members of the IUA as a collective mean to you?
4. Is there something the IUA could do that would benefit your company?

Given the economic climate and maturity of the product, what has worked in the past for the IUA doesn't seem to be working with today's version of the IUA.

So let me challenge each and every one of you to ponder these questions and work with fellow members in the message boards within the IUA/EIUA communities to offer constructive ideas, suggestions, and concerns.

A new message category has been created for this discussion. Enter the message board and go to "The IUA in 2011 - Community Revitalization "category. Four threads, one for each topic, have been created for your input. Please note, Gary will not be cross posting this to the IDMS-L so the only place to voice your input is within communities. To stay involved with the discussion, you may subscribe at the category level or subscribe to the individual topics of interest.

The board will be meeting in November and your input is vital to the direction we take the IUA in the coming years.

Editorial.. cont'd from page 1

Please note that a number of the threads are probably more relevant to the development side of the IT organization than to DBAs – so by all means please circulate copies of "IDMS Connections" to your analysts, designers and coders. While you're at it, please mention to those team members about the IUA/EIUA Community Message Board Forum specifically for Application Development issues at https://communities.ca.com/web/ca-idms-iua-eiua-global-user-community/message-board/

Remember that the purpose of the IUA is to share about both IDMS, the DBMS, and IDMS's developer tool kit (ADSO, OLQ, Culprit, DML programming in Assembler and Cobol, IDMS/SQL and so forth) – so by sharing your copies of "IDMS Connections" you are doing your part to help the IUA to meet it's primary objective.

And now – I am pleased to be able to present for your enjoyment and information - another issue of "IDMS Connections". Tom Hebert shows how IDMS can fit into a "Cloud Computing" environment, providing some lively industry commentary along the way. Also, many thanks to Chris Hoelscher, to CA for its support, and to the IDMS-L "posters" whose contributions are reflected in this issue.

That's all there is – because there is no more!

Gary Cherlet

mailto:gary.cherlet@sa.gov.au
Justice Technology Services (Department of Justice - South Australia)
IUA Board Member Responsible for IDMS Connections
President Australian IDMS Database User Group (OZIUA)

MAINFRAME CONTINUITY FOR CA IDMS

Like you, CA Technologies is also facing the challenge of Mainframe Continuity. It is imperative that we successfully train new software engineers to replace our mainframe workforce as they near retirement so that we can continue to provide high quality product support to you, and deliver innovation and value with new product releases.

In 2005, CA Technologies started a program for new Computer Science (or a related discipline) graduates in Europe. In 2009, this program was expanded to the United States. Our Prague-based European team continues to grow. In the U.S., each class in 2009, 2010, and 2011 added 20 new associate software engineers who were recruited from top Computer Science programs across the country. These new engineers are now working on various CA Technologies mainframe product teams.

The new U.S. associate software engineers spend their first two months attending our Mainframe Training Program in Plano, Texas. While there, they are introduced to the mainframe and learn that System z is the enterprise

(continued on page 4)

Mainframe Continuity.. cont'd from page 3

computing system that 90% of the Fortune 1000 companies rely on for their most important business needs. Topics covered in the Plano training include the z/OS operating system, the user interface for developing on the mainframe, an embedded UNIX-oriented operating system, programming in Assembler language, and the REXX scripting language. Along the way they also learn about popular mainframe transaction servers, databases, file structures, and change management systems. As part of their training, the new software engineers are introduced to the CA Technologies executives who run our Mainframe Business Unit, learning from them the tremendous benefits that CA Technologies customers realize from our mainframe software products.

After completing their basic training in Plano, the graduates go back to one of the CA Technologies main U.S. development centers (Plano, TX; Framingham, MA; Pittsburgh, PA; Lisle, IL; Ewing, NJ; Islandia, NY) where they immediately start working with their new product teams. Continuing the learning process within their respective teams includes working with senior software engineers on product issues and new releases. CA IDMS was one of the first product lines to benefit from this program when new engineers were hired in Prague in 2005 to join the CA IDMS product team. The new engineers learned quickly and are playing a significant role in the development and testing of new CA IDMS releases, which included CA IDMS and CA Visual DBA Versions 17.0 and 18.0.

The CA IDMS team also benefited when new U.S. software engineers from our 2009, 2010, and 2011 classes joined the team in our Framingham MA development center. Like their Prague counterparts, the U.S. software engineers have been quick to learn, and they played a key role in the development and testing of CA IDMS Version 18.0. Now, both the Prague and U.S. software engineers are part of the global team working with senior software engineers on the next CA IDMS release.

It is hard to believe that it has been over six years since we first started training new software engineers on the CA IDMS team–but the great results are proof positive. Our plans are to continue investing in new engineers to maintain the momentum we started in 2005 so we can continue to support you on CA IDMS well into the 21st century.

From the 'CA IDMS Management Team'

IUA/EIUA APPLICATION DEVELOPER FORUM URL:

https://communities.ca.com/web/ca-idms-iua-eiua-global-user-community/message-board/-/message_boards?_19_mbCategoryId=0&#p_19

Remember to subscribe at the Message Board level to get regular updates for all Message Board Categories, or just subscribe to one or more Categories of interest!

GETTING TO IDMS V18 WITHOUT WAITING FOR CA-MSM

Chris Hoelscher, Humana Inc.

Being the "eager beaver" that I am, and knowing that IDMS V18 will save our organization money, I have spent much time building a reliable and repeatable installation methodology without waiting for CA-MSM to be installed at our site, and I would like to share this methodology with the IDMS community.

The first question might be "why NOT wait for CA-MSM to Install IDMS V18? The first answer is that Humana does not yet have plans to have CA-MSM installed. The second answer is that, even if CA-MSM was installed, I would want to execute the install process manually before allowing an automated process perform the same tasks (this is akin to my 11th grade physics teacher who required us to pass a slide rule proficiency test before allowing us the use of hand-held calculators (keep in mind this *was* 1973).

However, not exploiting CA-MSM does not mean ignoring the CA-MSM methodology altogether – I patterned my install methodology after how (I think) CA-MSM operates; thus I separated my process into 4 phases: ACQUISITION (getting the software files on my mainframe), INSTALLATION (creating and populating software libraries), DEPLOYMENT (copying libraries from the "install" environment to the runtime environment(s)), and CONFIGURATION (customizing the software for our site usage (actually – I even split this into pre- and post- DEPLOYMENT phases, so perhaps there are actually 5 phases). So let's get to it. Following is a list of the steps I used to get my CA-IDMS V18 environment up and running.

Under the heading of the ACQUISITION phase, it was necessary to determine my UNIX root directory – I then need to run a job to create the needed UNIX subdirectories; in my case I ran:

```
MKDIR '/u/userid/autousers/myid/IDMSR180/' MODE(7,5,5)
OSHELL CHMOD 755 /u/userid/autousers/myid/IDMSR180/
OSHELL CHOWN –R D902DBA:CMH5342/u/userid/autousers/myid/IDMSR180/
```

Figure 1

Then, I went to the CAT support site, to the download center, selected products, selected CA IDMS/DB – MVS, 18.0 for the release, and clicked on GO – I was taken to another page where you I was presented with a list of objects to download – I selected **CA IDMS/DB PRODUCT PACKAGE**, made note of the **object name (I00000DOA00.pax.Z)**, and clicked **DOWNLOAD** . I was then taken to a **DOWNLOAD METHOD** page. I selected **FTP REQUEST** (I chose **FTP REQUEST** rather than **DOWNLOAD MANAGER** so that I can FTP directly to the Mainframe). I was sent an email informing me that the file has been placed for FTP. When I got the email(about 2 minutes after I selected **FTP REQUEST**) , I clicked on the provided link which took me to the **REVIEW DOWNLOAD REQUESTS** page, clicked on the Preferred FTP Instructions associated

(continued on page 5)

Getting to IDMS V18.. cont'd from page 4

with the correct order #, and took note of the **Host**, **Username**, **Password**, and **FTP Location** information. I exited the CA website.

I then FTP'd the base install file directly to my UNIX subdirectory as follows:

```
//FTPSTEP EXEC PGM=FTP,PARM='(EXIT=08'
//SYSPRINT DD SYSOUT=*
//OUTPUT DD SYSOUT=*
//INPUT DD *
HOST
USERNAME
PASSWORD
cd FTP LOCATION
binary
get object name +
  'directory path/object name' (replace
QUIT
```

Figure 2

I downloaded the documentation directly to the PC, as Mainframe bookmanager-format documentation was not created for IDMS v18 (not a happy day for me). Additionally, there were NO Product Authorization sheets to download as they no longer exist for IDMS V18.

Next, I unpacked the install object into constituent UNIX files:

```
//UNPAXDIR  EXEC PGM=BPXBATCH,
//         PARM='sh cd /directory path; pax      X <- col 80
//         -rvf object name'
//STDOUT  DD SYSOUT=*
//STDERR  DD SYSOUT=*
```

Figure 3

I then used invoked TSO ISHELL to edit the just-unpacked file UNZIPJCL – I made the required customization changes and submitted the job. We do not have ACS rules in place to direct placement of datasets based upon their HLQ, and this job does not have the ability to specify a STORCLAS (specifying VOLUME was ignored by our default SMS rules) , so I needed to move (using ADRDSSU) the created datasets to more appropriate volumes after completion of UNZIPJCL. This completed the ACQUISITION phase of the implementation process.

In the INSTALLATION phase, the steps looked somewhat (but not totally) familiar to what I have done in the past. First, the SAMPJCL dataset (or the dataset that contains member AGJSEDIT if it is copied) needed to be defined to my TSO session in the SYSPROC allocation – I did this by adding the following line in our TSO login script dataset member

```
%TOCONCAT SYSPROC HLQ.SAMPJCL
```

Figure 4

Next, I customized the AGJSEDIT member based on our site standards. A lesson I learned the hard way is the both DASDHLQ and SMPETEMP should be different from each other and different from for the values specified for GLOBALHLQ, CAIT0HLQ, PRODHLQ, and SAMPHLQ (which all were assigned the same value). Next ,I edited AGJ1ALL , at the command line typed AGJSEDIT and the customized changes auto-magically appear (the alternative would have been to make the changed by hand for each member to be submitted). This member, which allocates SMP/E, target, and distribution libraries, was then submitted. After successful execution of AGJ1ALL, I did the same with AGJ2CSI (allocate and format the CSI).

Before going further, it should be noted that in IDMS V18, ALL products (with the exception of a few optional interfaces) are installed, whether you intend (or are licensed) to use them. AGJ3RECD (SMP/E RECEIVE) is next up – I commented out the culprit interface FMIDs and RECEIVED the remaining 2 FMIDs (IDMS base and IDMS CICS support).DO NOT RUN AGJ3RECT if you are installing from the internet downloaded file – it is used only for a TAPE –based install. AGJ4APP (SMP/E APPLY) is next, with the same commenting (twice) required. AGJ5ACC (SMP/E ACCEPT), again with required commenting, completed the INSTALLATION phase.

It is certainly true that what I performed in **CUSTOMIZATION PT I** could be deferred to the first iteration of PT II but I chose to do as much as possible as early as possible to get that work out of the way. Having said that, following are the steps I followed.

1. In the ***HLQ*.CAGJSRC** dataset, I edited member CFIG#JCL, manually customized the changes I needed (this job was not created to utilize AS-GJEDIT), and submitted. This job creates and populates a new dataset**, *HLQ*. CAISAG.CONFIG.** This dataset contains the familiar jobs needed for the remainder CUSTOMIZATION phase.
2. Once ***HLQ*. CAISAG.CONFIG** has been created and populated by CFIG#JCL , customize the JOBCARD and SETUP members to meet your needs, Then it's time to edit your "old friend" (?) VARB-LIST. There are a few changes for this release:
   - α. The product selection area is slightly changed (YES vs INSTALL) (I guess because you've already installed them);
   - β. Some products have been rolled into others (TSO INTERFACE is no longer selectable, it is now an unconditional part of UCF, TP/CICS is no longer selectable, it is an unconditional part of IDMS CICS SUPPORT) and
   - χ. The TOOLS and ENDEAVOR products are now customizable as part of the base install, and there are 4 NEW datasets that are created as part of this process, the "CUSTOM" libraries:

(continued on page 6)

Getting to IDMS V18... cont'd from page 5

i. SRCLIB, kinda sorta like the old PPOPTION library,

ii. JCLLIB, kinda sorta like the old SAMPJCL library,

iii. LNKLIB (new; link edit parms), and

iv. LOADLIB (kinda sorta like, but in addition to, the old DBA.LOADLIB)

It should be noted at this point that for IDMS V18, CAT has reversed a 20-year approach (presumably to parallel CA-MSM), and no longer uses SMP/E to customize site-specific customizable load modules (this will come as a relief to many - I had spent much time over the past 20 years getting almost all my customizations installed under SMP/E).

I customized VARBLIST (much of it does not pertain to upgrading from a previous release) and ran CAISAG. I looked at the resulting members (JOBxx in my case) stored in ***HLQ*****. CAISAG.CONFIG,** and made changes to VBARBLIST where needed and reran CAISAG to my satisfaction.

JOB01 allocates the CUSTOM libraries – the sizes provided may or may not be suitable for your site, depending upon how many versions of source, JCL, or link parms you have. At this point I created additional libraries: I created custom loadlibs to parallel every runtime library that is used – this allows me to get the customized load modules into parallel libraries now, and simple perform a one-to-one move during DEPLOYMENT.

Additionally, the CAT process would have you add the CUSTOM.LOADLIB to your CV startup and batch jobs – I chose not to, opting instead to use the existing runtime (and parallel install) DBA.LOADLIBs for this purpose, already in our CV startups and batch jobs. JOB02 populates the source templates. JOB03 as delivered would assemble/link the source members into the CUTOM. LOADLIB –I chose instead to split JOB03 into many many jobs, one for each module to be customized, with multiple steps for each version of the customized module. Why?

I chose not to use the CUSTOM.LOADLIB library, I had many version of the source members that needed to be individually assembled (SRTT, SYSIDMS, SVC, CINT, INTC, etc) and I had IDMS system customizations for which no JCL was provided (SYSIDMS, RHDCFSTB, IDMSUXIT (new), ADS USERBIFs, FE Tables). In created JCL for customization where it was not provided, and separate versions of source, JCL, and link parms where it DID exist, and ran these in place of JOB03.

Some new modules to discuss:

- RHDCPINT creates a module to determine what products are intended to be used (LMP keys still determine what products may be legally used) – the contents of RHDCPINT can be seen in DCPROFIL – but not all products are displayed there (CICS Support is not displayed),
- WTOEXIT is now a standalone module and MUST be run from an authorized load library if z/IIP is desired – also – the delivered WTOEXIT now checks for a log full condition (DC050004) and need not be added as a customization,
- RHDCVEND does something but I am not sure what – it is delivered, we re-link it but do not change anything about it, so I am not sure on this one, and
- IDMSUXIT is the root anchor for system exits such as IDMSIOX2.

After much experimentation, I got all the source, JCL, and link parms correct, and got the load modules where I wanted them. This completed the CUSTOMIZATION PT I phase. Next, I had our system folks run JOB04 (CAIRIM), after, of, course, I had given the modules in ***HLQ*****.APFLIB** (including a new module CAIXDOA$) The usual caveats still apply as to how and when to run CAIRIM.

We move ahead in time to the day of implementation. I started with the DEPLOYMENT phase. The DEPLOYMENT phase was, in fact, the easiest to accomplish. I merely backed up all the runtime files that were about to be overlaid, and then copied the Install libraries (both CAT-provided and user created) to the corresponding runtime libraries. Then, on to CUSTOMIZATION PT II.

In **CUSTOMIZATION PT II**, I returned to ***HLQ*****. CAISAG.CONFIG.**

- JOB05 copies the DMCL/DBTABLE load modules (that own the dictionaries and catalogs) to the install environment for use in subsequent jobs).
- JOB06 creates JCL copybooks.
- JOB07 updates system thingies (protocols attributes messages). If each CV to be migrated has its own SYSTEM dictionary, run the appropriate steps multiple times.
- JOB08 was empty for me.
- JOB09 updates the SYSDIRL dictionary, updates SYSGEN components, and generates SYSTEM 99.
- JOB10 updates the APPLDICT dictionaries with system thingies. Again: multiple APPLDICTs, multiple executions.
- JOB11 and JOB12 were empty for me.
- JOB13 updates TOOLDICT entities.
- JOB14 executes a SYSGEN to reflect TOOL products.
- JOB15 was empty.
- JOB16 copies the SYSTEM 99 programs and tasks to SYSTEM 90. Repeat this job for each SYSTEM used by CVs being migrated.
- JOBs 17 and 18 were empty.

(continued on page 7)

SCALABLE CLOUD COMPUTING WITH IDMS

Introduction

Cloud Computing is the best thing to happen to IDMS in many years. I bet your organization has heard about Cloud Computing but I doubt its implications are fully understood as it relates to IDMS.

What if management decides they want to use software products like SAP, SalesForce.com and others who have embraced Service Oriented Architecture? What if management would really like to create new mission-critical, high-volume applications that take advantage of the architectures and technology inherent in Cloud Computing? What if you have IDMS? How will you get them there, intelligently, without throwing babies out with the bath water?

I can tell you this. Screen scraping isn't going to do it. The likelihood that your current ADS dialogs relate directly to the needs of the Cloud is near zero. Porting isn't going to do it either. Building an entirely new system on a new database platform and cutting over to it in one day is not feasible or it would likely have been done by now.

The way to move forward is to keep IDMS, allow current applications to exist for the duration of their economic life, and wrap IDMS with an architecture that allows you to use Cloud Computing to the benefit of your organization.

> *If there is no advantage to using relational databases on the Cloud then, there is no advantage to converting away from IDMS.*

Here is another bit of news. Relational is actually old and it is not a perfect choice for the Cloud. It is certainly no better than IDMS.

Getting to IDMS V18... cont'd from page 6

I then performed the following: Had our CICS folks update the CICS PPT with the input provided, format the LOGs and JOURNALs, remove the TOOLs loadlib from the Startup JCL, Started the CVs: Verify CICS access, TSO access, Batch access, and online access.

Additional thoughts: before CUSTOMIZATION PT II, I saved off any messages, modules, or subschemas (SYSDIRL only) that were either created or customized – I re-applied the customization or re-added them after the CUSTOMIZATION PT II.

I did not overlook the CAT-provided documentation; the install guide, best practices guide, and the new features guide (for the past several releases), and system operations guide were essential in making this happen correctly.

As I continue to refine this process as our environments are move to V18, look for updates on IDMS-L or in CONNECTIONS for updates, or you may contact me off-line at choelscher@humana.com.

In fact there is a nascent movement out there called "NoSQL", which does not literally mean no SQL but more closely means "Not only SQL". It has taken a while, but the exclusive, iron-clad grip that relational has held regarding data is beginning to loosen.

So let's explore Cloud Computing and IDMS. It could be just what the doctor ordered.

Cloud Computing vs. "The Cloud"

For this discussion, let's define the difference between "Cloud Computing" and "The Cloud." Cloud Computing is a design and implementation architecture. "The Cloud" refers to the collection of Cloud Computing Services available on the Internet.

We use Cloud Computing because we want to follow an architecture that allows us to safely invest in our applications. We expose services for internal and external use following the same architecture.

What is Cloud Computing?

Cloud Computing is more than putting up a web site that scrapes screens or reads from a copy of the IDMS data. Participation in Cloud computing brings new requirements and risks. Meeting these requirements and mitigating these risks can only be accomplished using a well-defined architecture.

Major Internet players and industry leaders are driving the evolution of the Internet from a collection of servers into a fabric of interconnected Cloud Computing services. It is only logical that private networks migrate toward the same architecture. It's driven by users and consumers, not the accountants and DBAs. Most users tend to think in terms of documents, for example, purchase order, sales order, bill of material, and work order. You create, file, and retrieve documents. You sort through them and aggregate their contents. It is natural that computer systems will evolve to a model more easily understood by the consumers.

Cloud Computing is interoperable. Any platform, operating system, or database can participate as long as it follows the rules and the architecture sets out the rules.

In Cloud Computing documents rule - not relational row sets. The old client/server row set-based world made us massage IDMS into a more limited format. We had no choice because virtually all client/server tools were built around the relational model. That was a disadvantage. But Clouds are built around the users, and the users want documents. This levels the playing field between IDMS and Relational. If there is no advantage to using relational databases on the Cloud then, there is no advantage to converting away from IDMS. New Cloud applications can coexist with existing ADS and COBOL applications and share the same IDMS databases.

When properly implemented, Cloud Computing represents an evolutionary leap forward in flexibility, security, scalability and reliability. We can no longer take comfort in the statement "Someday those guys in the client server world will figure out that mainframers know production computing better than anyone." The fact is

(continued on page 8)

Scalable Cloud Computing... cont'd from page 7

that Cloud Computing is every bit as reliable, manageable and useable as a mainframe, and it scales extremely well.

If you have used services like YouTube, Bing Maps, and Google maps, then you have used Cloud Computing. These are no longer merely web sites. Sure, Bing Maps is a web site but you also get Bing Maps data from other sites like hotel chains, airlines, and car rental companies. For example, Marriott does not transfer you to the Bing Maps web site in order to show you a hotel location. They call a web service provided by Microsoft and show you the results in their web page.

Exploring further, a Cloud is a collection of accessible services that provide services to clients. The services offer data in such a way that many different platforms can consume the data. The current method of choice for accomplishing this task is the Simple Object Access Protocol (SOAP) which underpins services in a Service Oriented Architecture, the core design model used in Cloud Computing. Cloud services hide their own implementation from their consumers. In other words, if you call Microsoft Bing Maps or Google Maps for geographic data, you have no idea how or where they are servicing your request. The Cloud resides in or on physical resources that host, maintain, and service the infrastructure required to deploy, execute, secure, and maintain the services. You don't need to know the details in order to use them.

You are not in the Cloud because you merely put a web site on the internet. Your ISP's email service is not a Cloud Service. SalesForce.com can be a Cloud service and so can Google Apps or Office 365. It depends on how you use them. You have to integrate them into your enterprise systems, making them part of the fabric that comprises your organization's Cloud. To do that you need to build and consume SOAP-based services.

Clouds don't have to be connected to, or limited by, the internet. You can develop private Clouds and take advantage of the design architecture. In fact, most organizations should keep most of their Cloud private and only interface to the outside world when necessary.

Requirements

If you are to offer and consume services in a Cloud, there are rules you will be forced to follow and requirements you will have to meet. Here are a few:

Clouds are multi-platform and heterogeneous. The Cloud is not an exclusive club. All operating systems are welcome, and the data flowing in and out must not be platform dependent. This is what makes XML an enabler of Cloud computing. The format of an integer is always the same in an XML document while it could be very different on the platforms sending and receiving the document.

The consumer of a service does not know or care how the service is implemented. It can be on UNIX, Linux, Windows, z/VSE, z/OS, or some other platform. You cannot expose your callers to a platform-specific nuance.

- It follows that today's Clouds use an XML document-style interface. Similar to a text editor, you open a document, make changes to a document and store a document. Security says whether or not you are allowed to do these things. You are not Cloud friendly if your client must call you three hundred times to get 300 rows of data. The client should be given a document in one call and that document would contain the 300 rows. To perform updates, the client changes the document and sends it in for update. The service updates the underlying database after it passes edits. This is the model we use to interface with IDMS.
- Currently "The Cloud" is based on TCP/IP because the internet is based on TCP/IP. However, there is no reason that a private Cloud implementation inside your company cannot use other network protocols if they are more efficient and appropriate to the situation.
- Clouds scale. Most definitions of Cloud computing require that service capacity be expandable by merely adding more machines. This type of scalability is how Google is able to service the massive number of search requests they receive each second.
- Clouds are reliable. Adding multiple machines increases reliability as long as the loss of a machine will not kick users out of the service. Sessions must be managed so that any particular call can be handled by the first available machine. Best practices for handling session data is moving towards cluster-wide cache services.
- Autonomous transactions rule. If you want reliability and scalability in your Cloud, you can't serve up database cursors to the clients. It's safe to say that merely exposing ODBC or JDBC from a server is not Cloud computing. Process an entire set of data in a single call, in a single transaction that is completed when the request is completed. This uncouples you from specific servers and allows the next request to be processed on a different server.

It is clear that building applications for the Cloud without forethought would be no better than building applications for IDMS with no forethought. Neither is a good idea. It follows that a well-thought-out architecture is needed.

Software Architecture

Cloud computing is gaining traction and it is showing up in more and more places. There are software and infrastructure architectures being developed to meet the requirements of Cloud Computing.

Current Best Practice is to use Software Layers such as Presentation, Service, Application, Business Rules and Data. The following is a greatly simplified diagram of one such possible architecture.

(continued on page 9)

Scalable Cloud Computing... cont'd from page 7

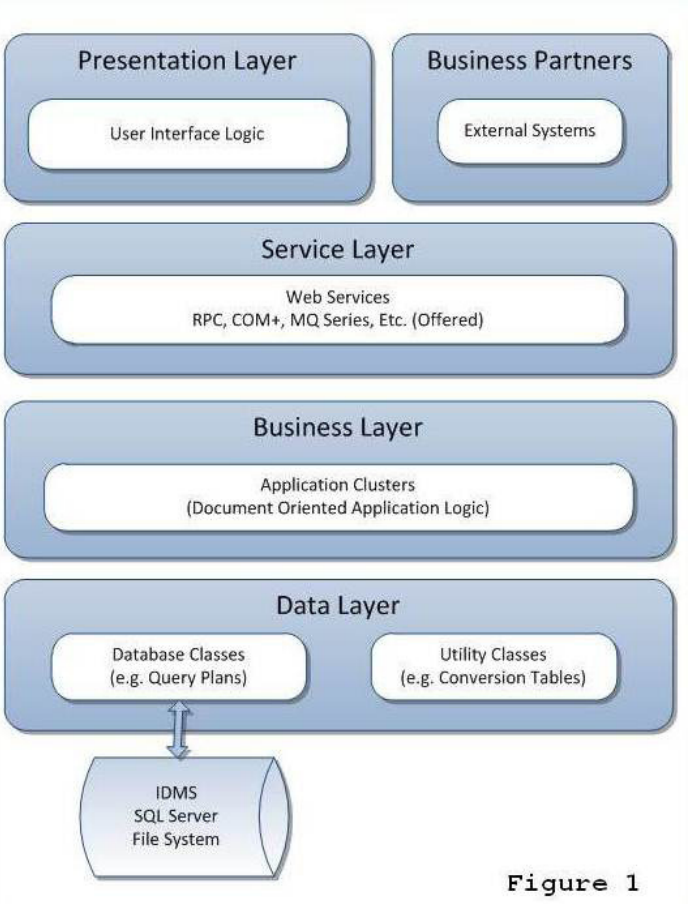


Figure 1

Calls generally flow from the upper layers to the lower ones. The Presentation Layers call the Service Layer. The Service Layer calls the Business Layer and so on. Layers are not normally bypassed. For example, it would be rare, but possible, for the Service Layer to call the Data Layer directly. That would occur only if the Business Layer has nothing to add.

But lower-level layers NEVER call higher level layers, nor do they perform functions meant for higher levels. For example, if the Service Layer were to call the Presentation Layer, how would it know what type of presentation facilities are available? It makes no sense.

Let's describe each layer from the bottom up.

Data Layer

Contains interface programs that retrieve and update data. Data sources could be databases, file systems, or even the service layer interfaces of other members in the Cloud (e.g., Google Maps). Programs in this layer could use IDMS SQL, XMLQuery, custom service programs (ADS dialog and TCPIP). You could even call your favorite screen scraper in this layer, if you don't care about scalability. One thing you don't do here is embed business rules in stored procedures. It destroys your ability to apply the same sets of business rules to multiple data sources.

Business Layer

This layer has a number of purposes. It calls various data sources using the data layer and massages the data for presentation in reports, web pages, and forms. This is a time-consuming function, as this is where transformation from a database row or record model into the document model needed by today's presentation options occurs. XMLQuery produces XML documents natively which greatly simplifies the task in this regard.

The business layer also takes in documents, and applies changes made to those documents to the underlying data sources. This is where edits are performed and database update methods are called. Object oriented techniques are often employed in this area after organizing the object design around the logical entity relationship structure of the business.

It is important to note that the closer you get to a "read the XML document," "change the XML document" and "store the XML document" model, the easier it will be for the presentation systems. This paradigm easily transports to web sites, forms, cell phones, pads, and many other devices. Placing all the edits and code transformations (edit and code tables) here avoids having them duplicated in the presentation layer.

Service Layer

The Service Layer facilities are generally used to allow consumers on one machine to call services on another. If all the layers are installed in the same machine, there is no need to call a service. The application could merely load and call the DLL or subroutine locally, unless caching is needed.

It follows that the Service layer is usually a simple gateway, allowing the Business layer methods to be called remotely across the Cloud's network. It is like a proxy server but with some extras. There are many occasions where caching XML documents can avoid repeated calls to the Data layer. Imagine, if you would, an application that presents the same data on various pages, or controls, in many different ways and you don't know which way the user will choose to see the data first. In this environment, each component in the Presentation layer will call the Business layer for a document, for example a purchase order. One may graph the data. Another may present it in a grid, and others as individual lines. You don't want each of these requests to hit your IDMS database every time. In that scenario you might use a caching facility in the Service layer. The caching service keeps XML documents in a rapidly accessible location (memory or local database) for a period of time. Therefore the first call to the Business layer would retrieve data from IDMS and store it in the cache before it returns the document. Subsequent requests would be fulfilled from the cache and not overburden your data source.

This solution may lead you to ask the question, "What if the data in IDMS changes while the item is in the cache? Wouldn't I get old data?" The answer is "yes," but there are a number of ways to deal with the situation. One way is to remove the items in the cache when they reach a certain age. Another is to refresh only when the user logs off. This way the user's data is static and consistent for the duration of their logon. A third method is to use triggers in the IDMS database to send messages to the cache and replace items in the cache automatically when they change in IDMS. If you think about it, is that not

(continued on page 10)

Scalable Cloud Computing... cont'd from page 9

what we have done with Data Warehouses and replicated databases? Finally, infrequently updated items would never expire. Edit and code tables are good examples.

At first glance the Service Layer seems simple. It's just a remote procedure call (RPC). However, it is the Service Layer that offers the key to performance, scalability and reliability.

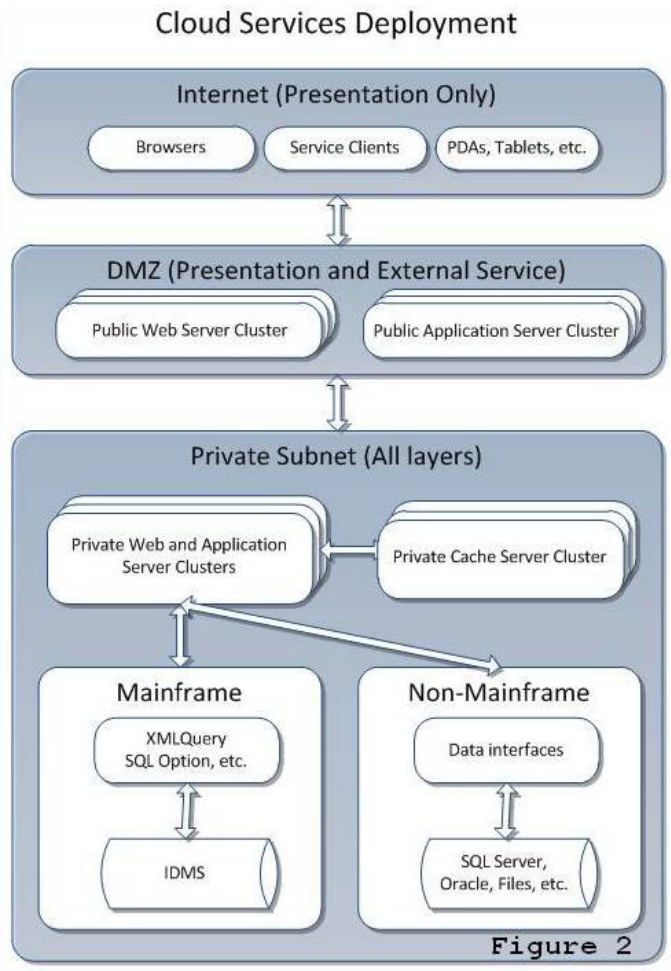
Presentation Layer

This is the layer that interfaces directly with application users. It can be Windows or Linux forms, web sites, personal data devices, cell phones, reporting subsystems and anything not yet invented. The Presentation Layer primarily interfaces with the Business Layer.

Reading data and calling other web services (like Google maps) from the Presentation Layer should be avoided. Implement these interfaces at lower levels so additional parts of your application can take advantage of them in the future.

Physical Deployment

The following diagram shows one option for deployment. There is no reference to platforms or operating system. Deployment can occur on any of, or any mix of, the following platforms: Mainframe, Linux, Windows, UNIX, and others.



Internet

The Internet needs little discussion as we all know it is an open network. It's the Wild West of computing. Be careful out there!

DMZ

The DMZ is a special subnet under your control offering web sites and Cloud Computing services for use by the public and/or trusted business partners and individuals. The machines in the DMZ usually have two network adapters: one facing the Internet and the other facing the DMZ. Because there is an adapter connected to the Internet, for security reasons it is a very good idea to deploy as little as possible to these machines.

I would strive to limit the DMZ to Presentation and Services - no business rules, limited caching of data, and NO DATABASES. You want to have so little in the DMZ that if a hacker were to break in there would be nothing for him to steal. Calls from the DMZ into the private network should be secured so that a miscreant user would not possess the credentials needed to call private subnet services.

You can see my first reference to "clusters" in the DMZ. Clusters facilitate scalability and reliability. Did you ever wonder how Google is able to accommodate the level of traffic and transactions that it handles with the response time it delivers? They do it using clusters of machines. Imagine hundreds or thousands of machines each taking requests in parallel. These machines are clones, each performing exactly the same function. There is usually a network device in front of a cluster that allows all the machines in the cluster to share the same IP address. Requests are passed to machines on a round-robin basis. If you find yourself short on capacity, merely add additional machines to the cluster. If a machine fails, it drops out of the cluster and the requests are handled by other machines. So there it is - scalability and reliability.

Private Subnet

The private subnet represents the networks inside your organization. Here you can find private clusters whose purpose is to perform the real work for requests coming from both the DMZ as well users inside your organization. This is where you locate your Data, Business and Inside Service Layers, and where you put your mission-critical applications and management systems.

Given the levels of demand anticipated in the Cloud, or in your private Cloud, it should be obvious that you cannot keep relational cursors open across conversations with users. IDMS DC can handle this using the TCPIP line driver, but how many PTERMs can an IDMS region handle, enough to support every Google user? I think not. For this reason data processing in the Cloud is autonomic. That basically means one call, one transaction. The only database sessions that exist are the ones being actively processed right now. Nothing is kept in the database region across conversations with users. That requirement leads us to use the document model.

The term for this concept is "sessionless." It's not just for Google. It applies to enterprise transaction processing as well.

The nice thing about Cloud Computing's document orientation is that documents are easily cached and easily retrieved from caches. It goes like this. A user signs into

(continued on page 11)

Scalable Cloud Computing... cont'd from page 10

their favorite mobile phone provider and asks to see call detail which will be retrieved from the database in the form of an XML document. The web site calls a service that first checks to see if a copy of the XML document is in the cache. If it is found in the cache, no database call is needed and the document is presented to the user. If it's not in the cache, the service retrieves the XML document from IDMS (in this case using XMLQuery), puts it into the cache, and then it is displayed to the user. The cached version of the document is available for a specified period of time, i.e. their logon session. So when the user asks to see the same data again, there is no need to call IDMS again.

The downside of web and application cluster usage is that each request from a given user may in fact run on a different machine making logon session management a challenge. This challenge can be addressed using enterprise caching mechanisms like nCache, Microsoft AppFabric Caching, Java Caching system, and others.

The way to move forward is to keep IDMS, allow existing applications to exist for the duration of their economic life and wrap IDMS with an architecture that allows you to use Cloud Computing to the benefit of your organization.

A cache server cluster makes objects in the cache available to all machines in the cluster, essentially improving the cache's scalability and reliability.

If the cache needs to be up-to-date at all times, a trigger loop can be created by combining a cache with IDMS triggers. When the triggers fire they extract the latest data from IDMS and update items in the cache. There are many ways to combine technical options for minimized demand on IDMS and maximized performance.

Summary

The advent, or evolution, of Cloud Computing has loosened the grip that relational databases have on data. That does not mean that relational is obsolete. It has strengths and those strengths will continue to be used. However, recent innovations surrounding IDMS have made it easier to write IDMS applications for Cloud Computing.

It now makes more sense than ever to keep your data in IDMS, yet proceed to take advantage of Cloud Computing. I will explore this topic in detail at CA World and hope to see you all there!

Author Biography

Tom Hebert began using IDMS as CIO at Purolator Products in 1982. He has since been a Director of Development at Cullinet, VP of Development at CA, a Management Consultant, and is currently the President of ObjEx Inc. Tom has expertise in both Cloud Computing design and development as well as deep experience in IDMS. He welcomes follow-up contact at tom.hebert@obj-ex.com or +1 (480) 588-7776.

Z/OS CPU EFFECTIVENESS: TIME USED AND TCB TIME EXPLANATION

The document TEC550739 is attached

Components:

CA IDMS/DB: 17.0
CA IDMS/DC: 17.0, 18.0

Last Modified Date: 22.07.2011

Document ID: TEC550739

Tech Document

Description:

This document includes information on the following topics:

How the CPU Effectiveness Values are Captured
Formula to calculate CPU effectiveness
CPU Time under TCBs
Values, Calculations, and CPU Usage
Example CPU Usage Calculation
Putting it all Together
I/O requests issued by IDMS
IDMS Wait Time

Solution:

For information on these topics please refer to link 'z/OS Time used and TCB Time'.

CPU Effectiveness

CPU effectiveness is the percentage comparison of CPU time to wall-clock time while the IDMS subtask (operating system TCB) was processing. For example, if IDMS is dispatched and uses one second of CPU time but it takes two seconds of wall-clock time to receive the one second of CPU time, the CPU effectiveness is 50 Percent.

As described in the IBM documentation and referenced in the following sections, there are certain interruptions that occur after a TCB is dispatched, but before it voluntarily gives up control. These interruptions result in the CPU being taken away from the address space currently active and instead executing instructions. The CPU time spent while handling these interruptions is not charged to the address space being interrupted.

The IBM System Management Facilities guide (SMF), discusses the use of the *TIMEUSED* macro to account for CPU time used by a TCB; IDMS utilizes this macro.

How the CPU Effectiveness Values are Captured

Consider a typical case where IDMS voluntarily gives up control of the CPU because it temporarily has no work to do. It is waiting for external events, such as the completion of a database I/O or a terminal operator hitting an Enter key. In this situation the following steps occur:

(continued on page 12)

CPU Effectiveness... cont'd from page 11

1. When an external event occurs, the IDMS system is dispatched by the operating system.

   One of the first instructions the IDMS system issues is a *STCK* instruction. This gives you the time of day (TOD clock) at the point you were dispatched. For this example, this is Time 1.

2. When the IDMS system has no additional work to process it gives control back to the operating system. One of the last functions the IDMS system does is issue another *STCK* instruction.

   This gives IDMS the time of day just before it gives control back to the operating system. Call this Time 2.

3. After giving up control to the operating system, the IDMS system is in a *wait* status. The *elapsed time* is the difference between time 1 and time 2.

   Note: IDMS might not have had control of the CPU during the entire time. For example, the operating system could have taken control due to some external interrupt, or some other higher-priority address space might have been given control even though IDMS had work to do.

4. While the IDMS system is dispatched, between 1 and 2, it requests from the operating system the CPU time it actually used.

   The SMF 30 record for this run was extracted and the values displayed SMF 30 record in the *Processor Account* section. The field *SMF30CPT* is the TCB time used by the IDMS TCB. In this case it is 30.35 seconds:

Formula to calculate CPU Effectiveness

Use the following formula to calculate CPU effectiveness:

TIMEUSED

CPU Effectiveness = STCK End – STCK Start x 100%

CPU Time under TCBs

The SMF 1.12 manual describes CPU time under TCBs as: "Times under TCBs that are included or excluded in CPU time fields of SMF records." It describes the included andexcluded time as: "Included TCB Times"

Timing values accumulated for the address space under TCB control include:

- Problem program time
- SVCs
- Lock spins encountered in an MP
- Environment
- EMS (emergency signals between CPUs) interrupt occurring within a lock spin
- Abend/Abterm
- User SPIE
- ESPIE exit processing.

Times excluded are: (These times are not reported in the SMF 30 record)

- External interrupt time
- Page fault processing time, including resolving page faults from expanded storage
- CPU stopped time if the QUIESCE command is used
- Attention processing time for TSO/E.

The following times are excluded but are available in the SMF 30 record):

- I/O interrupt time (accumulated separately in SMF30IIP)
- Swap-out/swap-in processing
- I/O error recovery processing (accumulated separately in SMF30RCT)
- Managing hyperspaces
- Program check handling.

Values, Calculations and CPU Usage

Consider you have the wall-clock time and the IDMS address space had work to do from the two STCK commands we previously mentioned, and then the TCB CPU time used from the TIMEUSED macro. Based on the IBM SMF documentation, the TIMEUSED value returned excludes specific times for the items such as Paging, Swapping, and external interrupts. Higher priority work can get control from an external interrupt.

Therefore, CPU effectiveness is the ratio of the CPU time used by the IDMS system to the time it took to get that amount of CPU. Theoretically, the CPU effectiveness could be 100 percent, meaning you are dispatched for (x) amount of wall-clock time and never had the CPU take over from IDMS due to an external interrupt. However, 100 percent effectiveness is not realistic.

To begin with, some time is not recorded, such as the time used during a shutdown process. This time is small in the overall scheme of things, but it is not accounted for.

The longer a system runs, the less significant this shutdown time becomes. Additionally, there will always be interrupts.

The main factor to consider when evaluating CPU effectiveness is the impact higher priority work has, as well as how Work Load Managed and LPAR balancing affect your processing.

Example CPU Usage Calculation

The following scenario demonstrates the CPU effectiveness values and how the CPU time displayed by IDMS, and used in the CPU effectiveness, can be cross-checked with SMF 30 step termination records, which are subtype 05. We took the values from the Processor Accounting Section of the SMF 30 record.

(continued on page 13)

CPU Effectiveness... cont'd from page 12

We started an IDMS system was started and executed a series of online transactions, using a network simulation tool. We issued DCMT commands and shut the system down.

The SMF 30 record for this run was extracted and the values displayed SMF 30 record in the Processor Account section. The field SMF30CPT is the TCB time used by the IDMS TCB. In this case it is 30.35 seconds:

```
********* RECORD 1   *********************************

SMF30CPT SMF30CPS SMF30ICU SMF30ISB SMF30JVU SMF30IVU  SMF30JVA
   30.35     3.73      .11      .00      .00      .00       .00
SMF30IVA  SMF30IST  SMF30IDT  SMF30IIP SMF30RCT SMF30HPT SMF30CSC
     .00       .00       .15     1.06      .00      .00       .00
SMF30ASR SMF30ENC SMF30DET SMF30CEP SMF30OFA SMF30EFA SMFDETOI
     .00       .00       .00       .00       .00       .00       .00
SMF3TIOC SMFETIOC SMDETIOC SMF4CEPI SMF3TOZ  SMFETOZ   SMFDTOZ
     .00       .00       .00       .00       .00       .00       .00
SMFTZOCP SMETZOCP SMDTZOCP SMETZQ  SMDTZQ
     .00       .00       .00       .00       .00       .00       .00
```

DCMT DISPLAY SUBTASK EFFECTINVESS output taken just prior to shut down.

Note: The value under Total CPU time for the TCB column. In this case, the value is 29.65 seconds. The reason for the difference between this value and the SMF30CPT field is the Shutdown time previously mentioned. This is the value returned from TIMEUSED and used in the CPU Effectiveness calculation.

Also note the value under Elapsed time for the TCB column. This is the time IDMS system was dispatched. In this case it is 45.19 seconds.

Subtask	Elapsed time		Total CPU time		%
Name	TCB	SRB	TCB S	RB	TCB
MAINTASK	00:00:45.1978	00:00:00.0000	00:00:29.6552	00:00:00.0000	65
Totals	00:00:45.1978	00:00:00.0000	00:00:29.6552	00:00:00.0000	65

The CPU Effectiveness is calculated by dividing the Total CPU time by the Elapsed time.

In this case: 29.65 / 45.19 = .65, which gives us a CPU effectiveness of 65 percent.

This example shows the IDMS address space was dispatched for 45 seconds to get the 29 seconds of CPU time it needed. In other words it took 16 seconds longer than it would have to process the workload, which was due to the external interrupts.

Going back to the SMF 30 record, you can see the SMF30IIP field accounted for 1.06 seconds of the 16 seconds lost due to external interrupts. The other fields as documented in the SMF manual are zero. The remaining 15 seconds lost are attributed to the items mentioned in the SMF manual as not being recorded. These are mentioned above and repeated here for ease of reference.

Times excluded (not reported in the SMF 30 record) are:

- External interrupt time
- Page fault processing time, including resolving page faults from expanded storage
- CPU stopped time if the QUIESCE command is used
- Attention processing time for TSO/E.

Putting it all together

For production systems, with respect to response time and throughput, anything above 90 percent is generally considered good. We have seen reports of CPU effectiveness in the high 90s. The real determining factor with respect to response time and throughput is meeting your Service Level Agreement (SLA). In other words, if you are meeting your SLAs and the CPU effectiveness value is 40 percent, this is probably acceptable. However, the SLA and effectiveness vary from site to site. The other point is a CPU effectiveness value of 40 percent means response time and throughput can be improved by changes that reduce the external interrupts such as higher priority work, paging, swapping, to name a few improvable items.

I/O requests issued by IDMS

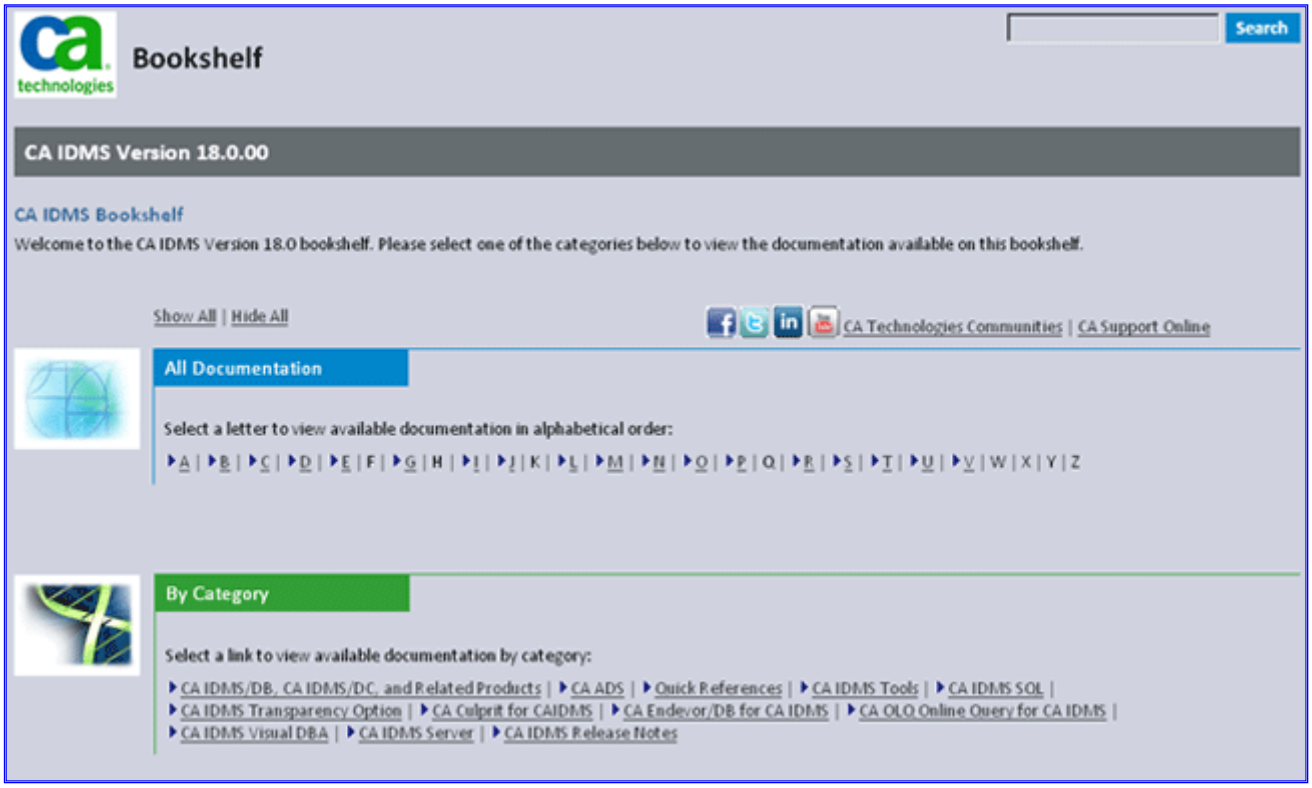
All database I/O operations within IDMS are done asynchronously. The IDMS system issues an I/O request and continues processing other work and tasks. It does not wait for the I/O to complete. Therefore, an I/O issued by IDMS does not affect the ELAPSED TIME field, with the exception for the minor amount noted in the discussion of I/O interrupts and SMF30IIP.

IDMS wait time

As we discuss in this document, IDMS processes I/O requests asynchronously. As long as some IDMS user task or IDMS system task—such as the Master task—has work to do, the IDMS system does not voluntarily give up control of the CPU. For example, user task X might need to wait for database I/O, but IDMS finds that user task Y can run. In that case, IDMS does not give up control of the CPU. However, there are times when every IDMS task is waiting for an external resource. For example, all user tasks might be waiting for database I/O and system tasks such as drivers are waiting for terminal input. At that point, the IDMS system voluntarily gives up control and enters a wait state. At some later point, one of external resources becomes available and the operating system wakes up the IDMS system. At that point, the IDMS system begins processing again.

The periods spent in a wait state are not relevant to this discussion of CPU effectiveness. No CPU time is changed against the IDMS system during this time, and this time is not included in the elapsed time shown in IDMS CPU effectiveness displays.

INTRODUCING THE CA IDMS HTML BOOKSHELF



Document ID: TEC550792

Tech Document

Title: CA IDMS Version 18.0 Documentation introduction

The HTML Bookshelf provides a single Help screen where you access the entire CA IDMS documentation set, with your choice of HTML and PDF formats. The HTML versions give you robust online viewing and search capabilities, while PDF versions provide you with a print-friendly option. This gives you instant access with the power to search the entire CA IDMS library from an easy, simple to use HTML interface.

Features

The CA IDMS HTML Bookshelf features include:

- A single help screen that displays all documentation for the Version 18.0.00 release.
- All books listed both alphabetically and by category. View all items, or limit your view to specific books or categories.
- An all-in-one search tool that searches the entire documentation set and returns matches found in both the HTML and PDF formatted documentation, without the need for a specialized .PDX index file.
- Additional links for using the bookshelf, downloading Acrobat Reader, and contacting CA.

Opening the HTML Bookshelf

You can view the HTML bookshelf online, or you can download to a local drive.

Note: You must have an appropriate license and registration to access CA Support Online.

To access the CA IDMS HTML Bookshelf

From any machine with Internet access, open the CA IDMS Bookshelf directly from CA Support Online: https://support.ca.com/cadocs/7/CA%20IDMS%20DB%20Version%2018%200%2000-ENU/Bookshelf.html.

You can stop here to view, search, and read, or continue the following steps to download the Bookshelf to a local drive.

To download the CA IDMS HTML Bookshelf

1. Locate and click the Download this Bookshelf hyperlink.
2. Follow the instructions to save the file to a local drive.
3. Extract the .zip file when the download is complete.
4. Double-click or run the bookself.html file to open the Bookshelf.

SEND ARTICLE CONTRIBUTIONS TO GARY CHERLET AT:

GARY.CHERLET@SA.GOV.AU

HELP DESK ARCHIVES

Backward Compatability Trivia Question

From: IDMS Public Discussion Forum [IDMS-L@LISTSERV.IUASSN.COM]; on behalf of; Rozeboom, Kay [DAS] [KAY.ROZEBOOM@IOWA.GOV]
Sent: Tue 11/09/2007 10:42 PM
Subject: Backward Compatability Trivia Question

After upgrading from R16 SP2 to R16 SP4, we had two ADS dialogs that were misbehaving. The problem was fixed by recompiling them under SP4.

The surprising part is that these dialogs had last been compiled in 1986!

They were still working fine last week, after 21 years, and 5 different releases of IDMS. Now that's backward compatibility.

Trivia question: What is your oldest ADS dialog load module, that is still being used in production?

Kay Rozeboom

State of Iowa
Information Technology Enterprise
Department of Administrative Services

Help > Desk > Archives

Record In Using List Not Used By Dialog

From: Agency Developer

To: DL: Help Desk

Subject: DC 177022 RECORD IN USING RECORD LIST NOT USED BY DIALOG

Hi,

Dialog GOF0026D displays a list of ORG-UNITs Offices for a user to select from.

If the user presses F3 function key the dialog abends with the following (From the editor: F3 is an ADSA Response that Invokes a Function mapped to a Cobol program – the Function definition has a list of "Using Records" that are known to the ADSA Application).

```
BROWSE -DIALOG GOF0026D HAS ABENDED..                    COLUMNS 001 079
COMMAND ===>                                             SCROLL ===> CSR
*** TOP OF DATA ************************************ CA-IDMS/ADSALIVE ***

ACCEPT DIALOG INTO WGR-DIALOG-NAME.
MOVE CURSOR-ROW TO WGR-CURSOR-ROW.
MOVE CURSOR-COLUMN TO WGR-CURSOR-COLUMN.
MOVE AGR-FUNC-DESCRIPTION TO WGR-STORE-
VALUE4.
MODIFY MAP TEMP CURSOR WGR-CURSOR-ROW WGR-CURSOR-
COLUMN.
MODIFY MAP PERM FOR ALL ERROR FIELDS EDIT IS
CORRECT.
IF AGR-NEXT-FUNCTION NE 'WHLPFL03'
THEN
 MOVE SPACES TO GUT01-SUB-MESSAGE.
DC177022 APPLICATION ABORTED. RECORD IN USING RECORD LIST NOT USED BY
DIALOG
EXECUTE NEXT
FUNCTION.
MOVE SPACES TO AGR-MAP-
RESPONSE.
*** BOTTOM OF DATA ********************************** CA-IDMS/ADSALIVE ***
```

This would indicate that the dialog is trying to move a value to a field in a record that is not known to the dialog. In this case GUT01-WORK-RECORD.

Cheers

To: Agency Developer

From: Help Desk

Subject: RE: DC 177022 RECORD IN USING RECORD LIST NOT USED BY DIALOG

The problem is not with the MOVE statement – if the record wasn't associated with the dialog it would not have compiled. The problem is actually with the EXEC NEXT FUNCTION. What is happening is that the EXEC results in a LINK to the DC-Cobol JIS Help program.

Because the dialog does not have one of the records used by that program associated with the dialog – the system is not able to pass addressability down to the Cobol program. Unfortunately the message does not tell us which record – so a little detective work is needed. This is probably a job for the ADS team. At the end of this e-mail is the message text associated with the DC177022 message from the display provided by Roland

HTH – cheers

MESSAGE ID DC177022

SEVERITY 1

DESTINATION LOG

MESSAGE TEXT APPLICATION ABORTED. RECORD IN USING RECORD LIST NOT USED BY DIALOG

DEFINITION DC177022 APPLICATION ABORTED. RECORD IN USING RECORD LIST NOT USED BY DIALOG

While processing an EXECUTE NEXT FUNCTION command, the system determined that the next function is a user program and that one of the records in the program's USING record list is not used by the dialog. Therefore, the link to the program cannot pass the record address.

A snap dump is generated.

Either recompile the dialog, adding the record as a work record or a subschema record or change the application so that passing this record to the program is not required.

Record In Using List Not Used By Dialog – End Game

This issue was followed up with CA on the basis that if the record is known to ADSA – then it doesn't matter whether or not the record is **also** associated with the Linking/Invoking dialog in the ADSA structure. The address of the record is known to ADSA and so ought to be able to passed to the Program Function in the ADS Application structure.

(continued on page 16)

Help Desk Archives... cont'd from page 15

Here is the resultant Apar from CA:

```
Apar TF24693
PRODUCT: CA IDMS/DB-MVS                          RELEASE: 17.0
PROB #:  4141_     FN: TF24693    FT: ABW        DATE:    6 JUN 2011
DC177022 ON LINK TO PROGRAM USING GLOBAL RECORD
```

ADS application abend 'DC177022 RECORD IN USING RECORD LIST NOT USED BY DIALOG' will occur when either execute next function or a global response is requested, but not every record needed by the LINKed-TO program is defined to the current dialog. The passed records must be defined to each dialog that might be current whenever the LINK to the program is requested.

This PTF will allow ADS runtime to pass ADSA-defined global records to ADSA-defined global Cobol responses.

SYMPTOMS:

An ADSA-defined Global Response that LINKs to a program passing ADSA-defined Global Records will abend with 'DC177022 RECORD IN USING RECORD LIST NOT USED BY DIALOG' whenever the passed record is not also defined to the current dialog or menu.

IMPACT:

Any dialog that LINKs to a Cobol program with a USING list of records will not not compile unless all records in the list are defined to the current dialog.

However, when a dialog LINKs to Cobol via execute.next. function, the passed record list is defined through ADSA so initial testing will not fail with DC177022 until that application function is tested.

The worst case is when a global Cobol function is added to an application to be initiated via a global response. Every dialog must be recompiled to add all records in the USING list. Menus will require coding an intermediate mapless dialog and a local response to initiate the globally accessible Cobol program.

CIRCUMVENTION:

Change every dialog defined in the application with the Cobol program as a 'valid response' to include all records that must be passed to the Cobol program.

Pre-requisites RO32382 and RO33169 and Apar TF24693

Help Desk Archives

CPU Increase

From: From X

To: IDMSVENDOR-L@LISTSERV.IUASSN.COM

Subject: CPU Increase

Hello All:

I assume expanding an area will result in increased runtime, I/O, and CPU time, especially if performing area sweeps.

Can anyone provide a percentage of increase that one would expect to see in runtime, I/O and CPU if we doubled the size of an area?

Is there a rule of thumb that can be used?

To: From X
From: IDMS-L@LISTSERV.IUASSN.COM
Subject: Re: CPU Increase
From X,

The answer is the infamous "it depends". Are you doing an EXPAND PAGE to just double the page size. Then the amount of I/O will be the same, the CPU should not noticeably change, same thing for the runtime. If you are doing an UNLOAD/RELOAD, keeping the same page size but double the number of pages then you should see twice the number of I/O.

This in turn will double the amount of CPU that is used to get those I/O and ince I/O is usually a large chuck of batch wall clock time, then your runtime is going to increase accordingly.

There are some other caveats. If you are double the size because the areas are almost full, then you probably also have a large amount of overflow records and possibly fragmented records.

In this case, if you do an EXPAND PAGE you will still have the extra I/O for the overflow and fragment records. If you do the UNLOAD/RELOAD you would hopefully remove all of the overflow and fragment records. Also, these comments really only apply to those programs doing area sweeps.

Another issue is that these same batch jobs are rarely as simple as sweeping a single area. You are usually sweeping one area and processing other records in other areas. So the increase in runtime, I/O, and CPU is proportional to just the time the program spends in the area sweep.

I am hoping I didn't simplify this too much, but you can see where this is a lot involved in this and no easy way to name a 'rule of thumb' that could be any where close to accurate.

If you are not really that interested in accuracy, just state if you are doubling the area in size you are doubling the runtimes, I/O, CPU , etc... :-)

Help Desk Archives

CPU Increase

To: From X
From: IDMS-L@LISTSERV.IUASSN.COM

Subject: Re: CPU Increase

Let me EXPAND a bit on previous comments, and UNLD some of my thoughts. This gets into the uglier details that were avoided, so those with delicate constitutions should move on to the next message in your inbox at this point.

First question is how much of the access to the AREA is via sweep, how much CALC/VIA? For a complex

(continued on page 17)

Help Desk Archives... cont'd from page 16

application/environment this is going to be difficult to determine.

Even for one program (as suggested) it's not a straightforward answer.

Here are the considerations for each type of access (either within a single program or global to the system):

For CALC/VIA/Index/DIRECT: as suggested, an unloaded/reloaded AREA could save some I/Os due to reduction in the amount of CALC overflows. Even for an XPAG, this will happen over time (as records come and go), but shouldn't show any immediate impact. For guesstimation purposes this can probably be ignored and this portion of the CPU/elapsed time/IOs can be considered constant.

For sweeps: the total amount of I/Os goes up, by whatever factor, based on change in number of pages. The elapsed time should keep in step with the I/O count. The CPU increase % will lag this, possibly by a large factor.

IDMS spends part of the cycles processing the request, part of the cycles doing I/O.

Since the same number of requests are being processed (until more records are added), the "processing" portion of the CPU should remain constant, while the CPU needed to perform th I/Os goes up proportionately to the size expansion. So depending on how dense the record populates the AREA, this portion is greater or lesser (the fewer records, the greater the percentage of CPU spent just doing I/Os).

The trick then is to somehow analyze a workload (single program or entire CV) and categorize it into these two components.

For a batch job, a product like STROBE might be able to supply sufficient data to allow a projection. Possibly even for a CV, although that would be messier.

Cheers

Help Desk Archives

IUA/EIUA User Contributed Library Has been seeing some action lately – here is the URL:

https://communities.ca.com/web/ca-idms-iua-eiua-global-user-community/document-library/-/document_library

Why not check it out?

Submit JCL to JESRDR

From: FromX

To: IDMS-L@LISTSERV.IUASSN.COM

Subject: DC COBOL question

I have a technical question I was hoping you could help me with... when using DC COBOL, what would be the top 3 methods (in your shop) for submitting a JCL to the JESRDR queue for processing? The DC COBOL program in question would be called by various ADS dialogues when it was found necessary to submit this job for further processing to occur.

We have an assembler program we use to LINK to from an ADS dialogue that will submit a job, but it doesnt seem to be working when we call it from DC COBOL.

Any and all suggestions would be warmly greeted and highly appreciated.

Regards

To: From X

From: IDMS-L@LISTSERV.IUASSN.COM

Subject: DC COBOL question

We use DC-Cobol for job submissions - used to use WRITE PRINTER statements to a "printer" which through JCL was pointed at INTRDR - the usual technique. We have since switched to using the CA-Spool provided API for dropping the JCL through there - avoids JCL images being written (and journalised) to DC-Queue as you do WRITE PRINTERs, then read and DELETED (and journalised) as the images are written to the printer. We have a lot of DC-Queue activity and getting rid of this overhead reduced deadlocking significantly. I can send you some sample code if you're interested. Having just the one "submitter" allowed us to make this a global change by modifying just the one DC-Cobol program.

HTH - cheers

From: From X

To: IDMS-L@LISTSERV.IUASSN.COM

Subject: DC COBOL question

Also, I would love some code samples, it sounds like what we would have done in CICS by writing to a TD queue identified with the internal reader.

Looking forward to hearing from ya!

Regards

To: From X
From: IDMS-L@LISTSERV.IUASSN.COM
Subject: DC COBOL question

Here's the sysgen and JCL bits and pieces that you'll need to provide the "plumbing" for flushing JCL statements out to the INTRDR:

(continued on page 18)

Help Desk Archives... cont'd from page 17

```
XXJISRDR   DD SYSOUT=(A,INTRDR),
XX         DCB=(RECFM=F,LRECL=80,BLKSIZE=80)

DIS SYS 1 WITHO ALL.
*+   ADD SYSTEM 1
*+       SYSTEM ID IS SYST0001
*+       .
DIS PTE JISRDR WITHO HIST.
*+   ADD PTERM JISRDR
*+       ENABLED
*+       IN LINE JISRDR
*+       MAXIMUM ERRORS IS 3
*+       PRINTER CLASS IS 1
*+       READBUFFER
*+       TYPE IS SYSOUTT
*+       PAGE LENGTH IS 60
*+       PAGE WIDTH IS 121
*+       .
DIS LTE JISRDR WITHO HIST.
*+   ADD LTERM JISRDR
*+       ENABLED
*+       PRINTER
*+           NOBANNER
*+           CHECKPOINT IS SYSTEM
*+           CONTROL IS SYSTEM
*+       PRIORITY IS 0
*+       PTERM IS
```

Here's the WRITE PRINTER statements (commented out because they are from the program that now uses the CA-Spool API instead of this method) - you will do this first one as many times as there are JCL statements to be submitted - whether they are passed to you in a work record, scratch area or whatever:

```
*******>>>>>>>>>WRITE PRINTER FROM W-JCL-LINE
*******>>>>>>>>>    LENGTH 80 DESTINATION 'JISRDR'
```

The following WRITE PRINTER you do once at the end - the '/*EOF' tells JES that is the last statement for the job and it flushes the buffer - otherwise you will have to wait until one or more jobs have been submitted to "flush" the earlier job out onto the INTRDR:

```
*******>>>>>>>>>MOVE '/*EOF' TO W-JCL-LINE
*******>>>>>>>>>WRITE PRINTER ENDRPT FROM W-JCL-LINE
*******>>>>>>>>>      LENGTH 80 DESTINATION 'JISRDR'
*******>>>>>>>>>COMMIT TASK ALL
```

If you're interested in the CA-Spool side of things let me know - HTH - cheers

Help Desk Archives

Writing to a queue from a COBOL subprogram

From: From X

To: IDMS-L@LISTSERV.IUASSN.COM

Subject: Re: Writing to a queue from a COBOL subprogram

Thanks for all the comments and suggestions so far. Some clarification:

- my subprogram does write a database record as well, and so I still need binds in my subprogram.
- I am currently not passing the subschema-control and extending the run unit to my subprogram
- this subprogram will be called from many different programs, so I am leery to do anything that will commit the data in the calling programs.

So if the FINISH TASK finishes all run-units, including the one from the calling program, does it also commit what was done in the calling program thus far?

To: From X

From: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: Writing to a queue from a COBOL subprogram

There have been many suggestions in this thread - one that I have not seen is the use of COMMIT ALL in the subprogram. COMMIT ALL only affects the one run unit - the ALL says to COMMIT both DB work AND Queue work.

This command will NOT impact any other run units associated with the batch task. Note that there is also a COMMIT ALL TASK which would impact other run units.

HTH – cheers

From: From X

To: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: Writing to a queue from a COBOL subprogram

I tried your suggestion with a COMMIT ALL instead of the FINISH TASK, and everything worked fine the first time I called my subprogram and when I returned from it, but the second time I called it I got a 5077 error code on the BIND TASK statement. So it seems like I still need a FINISH TASK in order to be able to BIND TASK again...

```
BIND TASK.
COPY IDMS SUBSCHEMA-BINDS.
MOVE 0 TO RETURN-CODE-E064.
READY INVLOG2-AREA USAGE-MODE IS UPDATE
    ON ANY-ERROR-STATUS
       MOVE 99 TO RETURN-CODE-E064.
READY IXINVLOG2-AREA USAGE-MODE IS UPDATE
    ON ANY-ERROR-STATUS
       MOVE 99 TO RETURN-CODE-E064.
READY IXINVDTE2-AREA USAGE-MODE IS UPDATE
    ON ANY-ERROR-STATUS
       MOVE 99 TO RETURN-CODE-E064.
PERFORM 2000-PROCESS.
COMMIT ALL.
FINISH.
```

(continued on page 19)

Help Desk Archives... cont'd from page 18

To: From X
From: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: Writing to a queue from a COBOL subprogram

I've made some suggested changes to the code you supplied. Note the removal of all "full stops" and the addition of the "end-if" after the "ON" statement - the "ON" generates an "IF" statement that needs a corresponding "end-if". Note also the MAJOR change to the code - to wit the

```
"if dml-sequence = 0 ....
.................
End-if" sequence.
```

This causes your program to only do the BIND-READY sequence the first time it is called. It will leave the Run Unit open for the next CALL thus giving you some performance improvements if it is called multiple times during the same batch job execution.

Now - about the open Run Unit because you now no longer execute a FINISH of any description - as long as the Cobol MAINLINE program - the one in the EXEC PROG statement in the JCL - issues a FINISH TASK - then all the Run Units left open in this way will be ended.

Here's the code:

```
MOVE 0 TO RETURN-CODE-E064

If dml-sequence = 0
   BIND TASK
   COPY IDMS SUBSCHEMA-BINDS
   READY INVLOG2-AREA USAGE-MODE IS UPDATE
    ON ANY-ERROR-STATUS
       MOVE 99 TO RETURN-CODE-E064
    End-if
 READY IXINVLOG2-AREA USAGE-MODE IS UPDATE
    ON ANY-ERROR-STATUS
       MOVE 99 TO RETURN-CODE-E064
    End-if
   READY IXINVDTE2-AREA USAGE-MODE IS UPDATE
    ON ANY-ERROR-STATUS
       MOVE 99 TO RETURN-CODE-E064
    End-if
End-if
PERFORM 2000-PROCESS
COMMIT ALL
.
```

Here is a bit of the techo stuff as I understand it - there was a major architectural change in Rel 12.0 - as previously each Run Unit did show up independently (and still does under the Run Unit screen - all with the same task code - but there is now only one task in the task code screen).

With the advent of the "mini CV" architecture a client-server relationship between the batch region (any external region) and the CV was implemented. Each external region connects on a single TCE/DCE pair - all the run units are represented by a chain of secondary LTE's that are created "on the fly" with each BIND.

I have written a "Run Unit" generator program that proves that I can have hundreds of Run Units hanging off of a single, batch TCE. This is demonstrated each time a batch task ends with one or more open Run Units - you see a whole lot of "DNS DTS client server" messages - I can't access the mainframe to show an example - this is all to do with the "client-server" relationship being broken in an unstructured manner.

HTH – cheers

Help Desk Archives

IDMS SQL DML related question

From: FromX
To: IDMS-L@LISTSERV.IUASSN.COM

Subject: IDMS SQL DML related question

Hi,

I am pretty new to IDMS world and I have a question. Can IDMSBCF through JCL allow me to do multiple row updates in a given record? If so would there be any cons to this method of update multiple rows? Our shop uses DMLO and other online tools. It seems they only use COBOL programs to do such updates otherwise it is DMLO (one by one).

The update query is of the following form:

```
Loop through all the Table1 records ##If the field CODE-TABLE= 'E' then
If the field CUSTOMER-STATUS = 'R ' then
Set the field CUSTOMER-STATUS to 'D '
```

Thanks,

To: FromX
From: IDMS-L@LISTSERV.IUASSN.COM
Subject: Re: IDMS SQL DML related question

Hi From X:

If you have the IDMS SQL option, and if you've created an SQL schema for your network schema, you could update the database with the following SQL query:

```
UPDATE table1
   SET CUSTOMER_STATUS = 'D'
   WHERE (CODE_TABLE = 'E' AND
                CUSTOMER_STATUS = 'R');
```

If you don't have the SQL option, you won't be able to use SQL against the database.

However, you do indicate you have DMLO. You could write a DMLO CLIST to loop through the database and do the updates. Please refer to the IDMS DML Online documentation or use HELP during a DMLO session.

Hope this helps!
Regards,

Help Desk Archives

(continued on page 20)

Help Desk Archives... cont'd from page 19

Linking Rules Extended and not with DC and Batch

From: FromX
To: IDMS-L@LISTSERV.IUASSN.COM

Subject: Linking Rules Extended and not with DC and Batch

Hi,

I was in a meeting here at the Navy Base concerning an on-line called program in an ADS environment which needed to be extended. Naturally, other related questions were raised, so I volunteered (I haven't learned yet) to write up an explanation.

Would anyone care to peruse the following and comment, correct, or whatever? Any insight would be appreciated, especially one that tells me my attempt actually already exists in an accurate form and where to find it.

BOLDS, ITALICS, INDENTIONS and the like have disappeared on the copy. I used some underline dashes to help.

Thank you in advance...

LINKING FROM NON-DC TO DC PROGRAMS AND VICE-VERSA CONSIDERATIONS

Definitions:

MODE: MODE statement in IDMS programs set to either; BATCH, IDMS-DC, or DC-BATCH

PA: Calling Program

PB: Called Program (that is, PA calls PB)

BATCH Pgm: any program running in an LPAR region started by JCL. MODE can only be BATCH, DC-BATCH, or non-existent. (no access to IDMS)

ON-LINE Pgm: any program running inside the IDMS-CV LPAR region, and started by a link or transfer from an ADS dialog or another program, or initiated by a TASK name in an IDMS session or a timer task. The program can also be an ADS dialog or non-IDMS.

Assumptions:

AUTOSTATUS appendage to the MODE statements will not be addressed in this paper.

BATCH Pgms

Linking When The Run Unit IS NOT Extended (That is, SUBSCHEMA-CONTROL IS NOT passed from PA to PB.)

PA and PB can each be any acceptable BATCH Pgm mode.

Linking When The Run Unit IS Extended

(That is, SUBSCHEMA-CONTROL IS passed from PA to PB.)

PA and PB must both be either BATCH or DC-BATCH

ON-LINE Pgms

Note: ADS will be treated separately at the end of each part..

Linking When The Run Unit IS NOT Extended

PA can be any acceptable ON-LINE Pgm mode. PB can be any callable program type.

When a program is called from ADS and issues a DC-RETURN statement to return back to the ADS environment or a DC-RETURN NEXT TASK CODE to navigate to another TASK Code, it must be MODE IDMS-DC to issue those statements.

The program can be BATCH if it calls another program which issued the DC-RETURN commands.

Linking When The Run Unit IS Extended

PA and PB must both be either IDMS-DC or BATCH.

When a program is called from ADS, it must be MODE IDMS-DC to accept the DC Communications block (SUBSCHEMA-CONTROL) from ADS.

Take Care

To: FromX
From IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: Linking Rules Extended and not with DC and Batch

I think that you are wrestling with a couple of issues here. **Firstly** - the MODE - defines the protocol that is used to access IDMS services. What the MODE does is directs the pre-compiler to generate different code to pass to the compiler.

In a BATCH environment the IDMS calls are to a module called IDMS, while DC-BATCH the calls go to IDMSDCBI and DC calls go to IDMSCOBI. Depending on the MODE different functionality is supported based on the environment that your program has indicated it will be running in.

MODE is BATCH does not support any but Database verbs, DC-BATCH supports all database verbs PLUS the WRITE PRINTER and QUEUE related commands. MODE is DC supports all of the above, plus SCRATCH, STORAGE, TERMINAL, MAPPING and other "DC" type functionality.

Secondly the issue that you are dealing with has to do with accessibility to SUBSCHEMA-CONTROL and already bound RUN UNITs. In order to have any communication with IDMS requires SUBSCHEMA-CONTROL - which is used to at a minimum pass the DB/DC verbs being invoked, other verb specific parameters, and receive status information back.

Whether or not a RUN UNIT is extended decides whether or not an existing, already BOUND SUBSCHEMA-CONTROL is passed to the CALLED

(continued on page 21)

Help Desk Archives... cont'd from page 20

program. In the case where an existing Run Unit is extended the CALLED program receives SUBSCHEMA-CONTROL in its LINKAGE SECTION.

In fact the CALLED program can chose to ignore this SUBSCHEMA-CONTROL (using a DUMMY place holder in the Procedure Division USING statement) and BIND its own RUN UNIT with a SUBSCHEMA-CONTROL that is in its WORKING STORAGE.

When there is no RUN UNIT extension - the CALLED program MUST establish its own SUBSCHEMA-CONTROL and issue the appropriate house keeping commands - BIND RUN UNIT and so forth.

It might issue the FINISH after each call - or if it will be CALLED multiple times it might do only the BIND on the first call and FINISH on the last call - or can rely on the mainline to do a FINISH TASK which explicitly finishes all bound RUN UNITS associated with the entire TASK.

Confused? So are lots of people - if there's anything I can do to clarify please let me know - cheers

Help Desk Archives

ADSO Area Readies – Brief Refresher

From: FromX
To: IDMS-L@LISTSERV.IUASSN.COM

Subject: COBOL exit programs vs. ADS mapless dialogs

Can anyone offer insight as to the pros and cons of using an ADS mapless dialog versus a COBOL exit program for commonly-used code that we want to call from several ADS dialogs?

Historically we have typically preferred using ADS mapless dialogs, as ADS has more built-in functions, etc. than does COBOL. However, often we find that later on we need the same processing done in batch, and often end up writing a COBOL batch version anyway. Are we better off just writing a COBOL exit program that we can call from ADS? Then we can easily clone and modify when we need a batch version. (We have started to do this more and more for any processing that we suspect we may need in batch as well at some point).

I understand that there are different run unit issues to consider with each. (With a COBOL exit, the ADS calling dialog's run unit will stay open while the COBOL program begins and ends its own run unit, unless we either do a commit before the call or code the dialog and COBOL program so as to specifically extend the run unit; whereas with a mapless dialog, the ADS calling dialog's run unit either finishes when we call the mapless dialog which then starts its own run unit, or extends its run unit to the mapless dialog depending on the subschemas and ready usage modes, whether we specifically extend it, etc.)

We also have some "include modules" that we sometimes use for commonly-used code, but we find the downside to those are that a change requires regenning of all the calling dialogs, resulting in extra work for us and more interruption for our users.

We have some processing that is commonly and frequently used throughout our application. Are there performance considerations to doing it in one way vs. the other?

Any feedback would be most appreciated.

To: FromX
From: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: COBOL exit programs vs. ADS mapless dialogs

FromX - I have some detailed stats on this very subject - they include performance for iterating within the "called" program from 10 to 10,000 times on a single "call" - and for iterating once inside the "called" module which was "called" 10 thru 10,000 times. Some very interesting results - at the end of the day the answer will be - "it all depends upon a great many variables". When I have a chance I'll send a detailed reply complete with graphs, etc.

'til then - cheers

From: FromX

To: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: COBOL exit programs vs. ADS mapless dialogs

Thanks very much for the information...very interesting stuff, and different than what I would have expected. Oddly enough, I would have expected ADS to be a more efficient language than COBOL (I guess I assumed that newer is always better and faster, which apparently isn't always the case).

I am still confused about one issue with COBOL exits though. My understanding is as follows...when a dialog updates a record, it does an exclusive lock on that record and those directly associated with it. This means that no one else can even read that record until the lock is released by a COMMIT ALL or by the run unit ending. When a dialog calls a COBOL exit, unless it passes the subschema-control parameter, the COBOL exit will open a new run unit, and the dialog's run unit remains open.

So if a dialog updates a record, then calls a COBOL exit without passing subschema-control, and that COBOL program reads the record that has just been updated, I would expect the COBOL program to deadlock with the calling dialog, yet we have places where it does exactly that and doesn't deadlock. (It does if the COBOL program tries to update the same record). Am I wrong in that the lock prevents the record from being read by another run unit?

Thanks very much again for taking the time and trouble to share your information with me.

(continued on page 22)

Help Desk Archives... cont'd from page 21

To: FromX
From: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: COBOL exit programs vs. ADS mapless dialogs

With regard to your "not deadlocking" issue - hmmmm - I need to speculate here. If the Cobol program readies in RETRIEVAL and the system generation says RETRIEVAL NOLOCK - then I suspect that you would see the behaviour that you have observed. Ready in an update mode or change the system generation parameter and I would expect the deadlock scenario.

I'd be interested to know the READY usage mode and whether you have RETRIEVAL LOCK or NOLOCK - just for my own information.

I forwarded my benchmark results to Greg Beedy at CA - just in case comments about ADS vs. Cobol performance start coming out on IDMS-L he will at least be aware of a possible source for the information. Early next year I'll re-run the benchmark on z/OS 1.9 and Release 16 SP(3).

Take care - cheers - Gary

From: FromX
To: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: COBOL exit programs vs. ADS mapless dialogs

Your hunch is correct, and makes a lot of sense. As you suspected, our SYSGEN statement is set to retrieval nolock, and the ready usage mode in the cobol exits where I've noticed this happening is retrieval. I've since read up in the documentation, and it definitely sounds like it is our scenario. Thanks for steering me in that direction.

One question about your benchmark COBOL vs. ADS results... did you do any tests where the cobol program/ mapless dialog did a lot of database I/O? I wonder if ADS does that more efficiently?

Many thanks again for all your information.....

To: FromX
From: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: COBOL exit programs vs. ADS mapless dialogs

Thanks for letting me know about your sysgen settings - it's one thing to know something theoretically - it's good when that's how it actually works in practice.

Cheers

To: FromX

From: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: COBOL exit programs vs. ADS mapless dialogs

Sorry - I missed the "follow on question":

One question about your benchmark COBOL vs. ADS results...did you do any tests where the Cobol program/ mapless dialog did a lot of database I/O? I wonder if ADS does that more efficiently?

No - I didn't bother benchmarking Cobol vs. ADS database retrievals - although I did benchmarks with regards to a main dialog doing database work - then LINKing to Cobol (one test) ADS (another test) - where the sub-routine also does some database work - both tests repeated for extending and not extending run units. From an efficiency point of view - avoid currency saves, use the smallest possible subschema, and extend run units whenever you can.

As for the relative efficiency of accessing the database - you need to remember that ADS does extra work on every database call - depending on your sysgen settings (re: retrieval NOLOCK) and your ADSC dialog compile options (see the "Retrieval locks are kept" option - default is turned ON - if you turn OFF you can reduce this overhead). So even on the database accessing front - Cobol is likely to win in many instances.

HTH - cheers

Help Desk Archives

ADSO Area Readies – Brief Refresher

From: From X
To: IDMS-L@LISTSERV.IUASSN.COM

Subject: Adso ready of areas.

I need a refresher. I know that ADSO dialogs do not explicitly code ready areas and do binds like Cobol programs do. The question is, which areas does ADSO actually ready? All the areas in the subschema that a dialog got compiled with? Or does ADSO look at the records (and thus areas) that are actually referenced in the code and the ADS compile? Is there a ptf that influences this behaviour optionally?

To: From X

From: IDMS-L@LISTSERV.IUASSN.COM

Subject: RE: Adso ready of areas.

I believe that there is one way to explicitly control the otherwise implicit behaviour of the compiler's creation of the Ready Area Table (RAT):

- Code READY ALL NOREADY
- Code individual READY statements for each required AREA and USAGE MODE

As pointed out in other replies - it does not matter where in the code the READY statements are coded - beginning, end, or middle - they are "compiler directives" and determine the contents of the RAT.

Also, as pointed out, the READY statements are only executed by ADS at run time when the first executable

(continued on page 23)

Help Desk Archives... cont'd from page 22

DML verb is encountered.

This can be confusing at times - since a LINK ... USING (.... SUBSCHEMA-CONTROL ...) will also cause a Run Unit to be bound (if there is not already a bound Run Unit) and the READYs to execute so that the Subschema Control for a bound Run Unit can be passed in the LINK.

If "NO" READY statements are coded then all of the Areas in the Subschema are Readied with the Subschema default or explicit usage modes. Each explicit READY in the ADS code only serves to override this default mode.

Just as you can code READY ALL NOREADY - you can code READY ALL Usage-Mode to explicitly set the Usage for ALL areas in the Subschema to the specified mode.

When you use a READY ALLL compiler directive - the compiler creates a single entry in the RAT that has an "ALL" flag as well as the nominated usage Mode.

In the case of READY ALL NOREADY the RAT contains a single entry indicating NOREADY, and then an entry for each explicitly code Area Ready.

HTH

Help Desk Archives

DC-COBOL Link to a ADSO Dialog

From: From X
To: IDMS-L@LISTSERV.IUASSN.COM

Subject: DC-COBOL Link to a ADSO Dialog

Good Morning all,

Has anyone successfully done a link to an ADSO dialog from an dc-cobol?

Thanks …….

To: From X
From: IDMS-L@LISTSERV.IUASSN.COM
Subject: RE: DC-COBOL Link to a ADSO Dialog

A couple of approaches that would let you use ADS code from DC-Cobol:

1) Using TCP/IP - set up a port with the supplied generic listener and associate it with a task code that is the name of a mainline ADS dialog and invokes ADSORUN1 - the mainline dialog can have the TCP/IP "smarts" and act as a "wrapper" for the existing code so it doesn't need to be modified - have the DC-Cobol program connect to the port using TCP/IP. With this approach the code is also usable from Java and .Net applications, as well as programs on other mainframes, CICS applications and so forth - anybody who can use TCP/IP can access the code.

2) "Wrap" the existing code with an ADS mainline dialog that is invoked (i.e. "called") as an SQL "called Procedure" - the DC-Cobol program can then use an SQL "call" or "select" to activate the ADS code - with this approach and with CA-Server in place the code is also usable from Java and .Net applications, as well as batch and online programs.

Just a thought - neither solution is a direct "call" solution but they both give you access to the code through industry standard interfaces.

HTH - cheers

From: From X
To: 'IDMS Public Discussion Forum'
Subject: RE: DC-COBOL Link to a ADSO Dialog

Thanks for all the responses on this. I opened an issue with CA, they say you can link from a dc-cobol to an ads dialog.

I want to thank Randy and Daphne from CA Tech Support for their help.

Here are the instructions from level 2:

It should be possible for a cobol program to LINK to an ADS dialog.

The cobol program must include:

```
WORKING-STORAGE SECTION.
   COPY IDMS UNIVERSAL-COMMUNICATIONS-ELEMENT VERSION 2.
```

and

```
PROCEDURE DIVISION.
       MOVE 'UMBR' TO UCE-IDENT-02.
       MOVE 'dialog name' TO UCE-ACTIVE-TASK-02.
       TRANSFER TO 'ADSORUN1' RETURN
            USING UNIVERSAL-COMMUNICATIONS-ELEMENT.
```

The dialog must also be compiled with version 2 of record UNIVERSAL-COMMUNICATIONS-ELEMENT if data will be passed within it.

Data can also be passed between cobol and ADS in the other fields in the UNIVERSAL-COMMUNICATIONS-ELEMENT record, or scratch, or queue, etc., but neither passed database records nor SSCTRL can be used by the dialog.

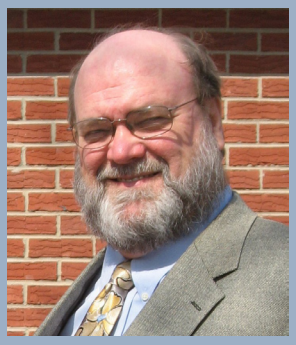
If this is a cobol menu LINKing to an existing ADS application, there will be minimal interaction between the cobol program and ADS and implementation problems will be minimal.

If you are adding an existing ADS mapless dialog as a subroutine to an existing cobol program, the interaction between the pieces in possible deadlocks, currency errors after the ADS @FINISH TASK, queue out-of-sync errors after a rollback, etc. could be complex.

Thanks again …….

Lesson learned: when you don't get the answer you want the first time – keep on asking!

International Chair
Company: Managing Member
Run Right, LLC
Email: lindajcasey@runrightllc.com



Secretary/Treasurer
Email Coordinator
Bob Wiklund
Company: Tiburon Technologies
Address: 17101 W. Gable End Lane,
Surprise, AZ 85387
Phone: 623 594-6022
Email: bob_wiklund@tiburontech.com

International Vice Chair
Contributed Software Librarian
Laura Rochon
Company: Ajilon Professional Services
Address: 22 Jolliet, St-Bruno,
Quebec J3V 4Z1 Canada
Phone: 514-943-8290
Fax: 450 441-6880
Email: l.rochon@videotron.ca

European IUA Representative
Steve Rundle
Company: British Telecom BT Group plc.
Address: PP2B33 Angel Centre,
403 St. John Street, London
EC1V 4PL UK
Phone: +44 (0)20 7777 6920
Fax: +44 (0)20 7777 6921
Email: steve.rundle@bt.com

Board Member
Terry Schwartz
Company: Perot Systems
Address: PO Box 269005
Phone: 972 577-3722
Email: terry.Schwartz@ps.net

Board Member
Craig McGregor
Axciom
craig.mcgregor@acxiom.com

Board Member
Diane Montstream
Allen Systems Group
diane.montstream@asg.com

Board Member
Jan Rabaut
jan.rabaut@dexia.be

Editor
Gary Cherlet
Justice Technology Services
South Australian Department of Justice
gary.cherlet@sa.gov.au

Desktop Publishing
Rebecca Shaw 404 247 8269
shawrh@bellsouth.net

IDMS Connections is a bi-annual publication of the CA-IDMS Database and Applications User Association (IUA). It is designed to promote its members' objectives. *IDMS 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
Web: http://iuassn.org