

DB2 for z/OS Version 8: Upgrade Planning

- *What We've Learned*

Tuesday, September 25, 2007

Prepared By:

William J. Favero
Senior Certified IT Software Specialist
DB2 for z/OS Specialty SSR
IBM Sales & Distribution
West Region

Edited By:

Roger L Miller
DB2 Architect and Lead Strategist
DB2 for z/OS Development
IBM Silicon Valley Lab

Table of Contents

Introduction.....	4
Reference material	5
Before the Product Tapes Arrive	6
Hardware Prerequisites	6
Operating System Prerequisites	6
Migration/Fallback APAR	7
Validate the DB2 Catalog	7
DB2 Tools Inventory	8
Running the IVP	8
DB2 Catalog Cleanup and Reorganization.....	9
Stored Procedures	9
Plans/Packages	10
The Question of Load Libraries	11
What About Compilers.....	11
Create a Project Plan and Build a Schedule.....	13
Version 8 affects on DSNZPARM	13
There could be a ZPARM out there just waiting to ruin your V8 upgrade	15
Planning Wrap-up.....	17
DB2 for z/OS Version 8 Utilities use of DFSORT	18
DB2 Version 8 DFSORT informational APARs	21
DB2 for z/OS Version 8 Reserved Words	21
SYSOPR	22
The report of “excessive CPU consumption” was an exaggeration	23
Compatibility Mode (CM) Overview.....	24
CM Coexistence	26
You’re in V8 CM and don’t like your access path. Now what?	28
Compatibility Mode – How long is long enough.....	30
Plans, Packages and DBDs in DB2 Version 8.....	32
Rebind after upgrading to DB2 V8.....	32
DSNTEP4	34
DSNTIAUL	35

Caution: SPT (SCT) Directory Table Space Size Increase	35
How do you make a DB2 DBD dizzy?	35
More on BIND: This time in regards to V8 NFM.....	37
Enabling New Function Mode (ENFM) Overview	38
Let's revisit ENFM for a minute.....	41
DB2 Version 8 and Unicode.....	42
What is Unicode?.....	42
Should Unicode be of concern?	43
Unicode Resources	45
Unicode and IFCIDs	45
CCSIDs, Unicode, and DB2 V7, Oh no!	46
Enabling New Function Mode (ENFM).....	47
DB2 for z/OS Version 8 Long Names	47
New Function Mode (NFM) Overview	49
NFM is running, now what?	50
Why move to V8 NFM.....	52
The secrets to a successful upgrade to DB2 for z/OS Version 8	53
What's your DB2 for z/OS V8 upgrade strategy?	55
Countdown to the end of DB2's Private Protocol.....	57
What's your incentive?	59
Migration/Fallback Info APAR for DB2 9	60
About the Author	61
Copyright Notice.....	62

Introduction

It has been more than three years since DB2 for z/OS Version 8 became generally available. Since that time we have learned quite a bit from some of the initial installations. One of the significant lessons learned is planning is **essential** to an upgrade's success. The better your migration plan and the more you understand what happens during the migration phases, the more successful your migration is going to be. This paper is intended as a tool to get your migration planning moving in the correct direction. Hopefully it will answer some of your questions and help you come up with some new ones.

Before you can plan a trip, you have to have a destination in mind. A migration is not all that different. Before putting together a migration plan, you should know something about the route and where you will eventually end up. The route to V8 is very interesting because it is new and different while remaining very much the same process as previous releases.

When you first hear the upgrade process to V8 explained, it does sound like IBM has come up with a new and possibly confusing way to get from V7 to V8. This is actually the furthest thing from the truth. The process a customer follows during migration is not all that different from the best practice process they have been following since DB2 Version 1. The only difference: the process has now been formalized. In the past, you updated the DB2 catalog using CATMAINT, tested the newly migrated catalog, tested the new DB2 code, and then enabled your users to the new features and functions being delivered in the new release or version. As you will see, that is the same process you will follow when moving to Version 8. First you will upgrade the DB2 catalog using CATMAINT and perform all initial testing of the new code just like in the past. Only now that step is called Compatibility Mode (CM) and new SQL function (plus a few other features) is not available. CM is also the only time you can fall back to Version 7. We apologize for the name Compatibility Mode, as this mode is not compatible. We have tried to make CM a conversion mode to make the transition as simple as possible. We introduce incompatible changes at this point, so that you can fall back to V7 easily. Preventing most new function is for your own good. This prevents a feature or function from being used during migration and possibly preventing your fall back to V7. Once you are comfortable that the catalog migration works and the new version behaves as advertised, you next want to convert the catalog to Unicode and enable the use of long names. This step is new and a little different than previous releases of DB2. It's called enabling new function mode (ENFM). Once you move to ENFM, there is no longer any falling back to Version 7, there is only "falling" forward. Finally, you want to actually start to use the new features being delivered by the new release. New Function Mode (NFM) is the name given to this step. NFM allows the use of all features (well just about all features) in Version 8. You can now move safely back and forth between ENFM and NFM modes, enabling and disabling the use of new features and functions.

That's the really quick 60,000 foot look at upgrading to DB2 or z/OS Version 8. Is that really all there is to it? Well, maybe not. There is a touch more involved in each step. We'll get into more detail, starting with a discussion of Compatibility Mode (CM) in just a minute.

If you need information about DB2 Version 8 and you are looking for "one stop shopping", there are some excellent materials covering planning, installing, migrating,

administrating, testing, etc... on IBM's DB2 website. Just about anything and everything you would want to know about DB2 for z/OS Version 8, broken down and organized by category, can be found at URL

www.ibm.com/software/data/db2/zos/roadmap.html.

Reference material

While on the subject of reference material, this seems like a good time to mention some other resources available. Too often, technical discussions end with references to additional details about whatever the subject happened to be. The problem with this, in my opinion, is that a lot of that reference material is as important, if not more important in some cases, as the discussion itself. Migrating to DB2 for z/OS Version 8 is such a situation. Because of that, there will be a discussion of a few pieces of reference materials right up front so the questions about "where do I find it" are answered from the get-go.

Of course, the most important DB2 related web site to bookmark is the IBM DB2 for z/OS home page at URL

www.ibm.com/software/data/db2/zos/

From here you can get to almost any information made available by IBM about DB2 for z/OS. This page is continuously changing as new stuff is made available, so plan on visiting often to stay up-to-date.

Probably one of my most frequently visited DB2 web pages is the IBM DB2 Support page. Located at www.ibm.com/software/data/db2/zos/support.html, it gives one the ability to search on just about anything DB2 related. It's a very easy and straightforward way to locate white papers, articles, Redbooks, presentations given by IBM, answers to questions, and information about PTFs and APARs.

No discussion about DB2 materials is complete without a list of current Redbooks. A few that I think are invaluable to anyone working with Version 8 are:

SG24-6763	The Business Value of DB2 UDB for z/OS
SG24-6489	Best Practices for SAP Business Information Warehouse on DB2 for z/OS V8
SG24-6480-01	Securiting DB2 and Implementing MLS on z/OS
SG24-6465	DB2 UDB for z/OS Version 8 Performance Topics *
SG24-6319	DB2 for z/OS and WebSphere: The Perfect Couple
SG24-6370	Disaster Recovery with DB2 UDB for z/OS
SG24-6079	DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More *
SG24-7088	DB2 UDB for z/OS V8: Through the Looking Glass and What SAP Found There
SG24-7083	DB2 for z/OS Stored Procedures: Through the CALL and Beyond
SG24-6418	DB2 for z/OS and OS/390 : Squeezing the Most Out of Dynamic SQL
SG24-7111	Data Integrity with DB2 for z/OS
REDP-4187	Disk storage access with DB2 for z/OS

* Must have Redbooks

A critical web site to your success with DB2 for z/OS Version 8 is the page containing all of the DB2 V8 reference manuals and product documentation. Located at

www.ibm.com/software/data/db2/zos/v8books.html, you will find the latest versions of all of the Version 8 product manuals, program directories, and extender documentation. The only manuals missing from this web page are the licensed materials. Most of the DB2 product manuals were last updated in February 2007. This is another one of those pages you want to bookmark so you can always check back to make sure you are referencing the latest and greatest DB2 information. You should be very careful whenever using hardcopy versions of the product reference manuals because you never know how old they are or what may have just been changed. Always check the web for the "new stuff".

Before the Product Tapes Arrive

Let's start with a migration preparation discussion before actually explaining the different migration modes in Version 8. You've just arrived back from IDUG, the DB2 Tech Conference, SHARE, or maybe a local database user group where you have been all pumped up about moving to DB2 for z/OS Version 8. What can you do while the decision is being made about when to order the new DB2 and when to install it? Maybe you can use this time to do a little analysis and planning and get ready for your inevitable migration to Version 8?

Hardware Prerequisites

One of the first things to determine is if you are properly positioned to even consider a V8 installation. DB2 Version 8 has some very specific hardware and software prerequisites because of its exploitation of 64 bit architecture. DB2 for z/OS Version 8 has a hard requirement for a processor that supports zArchitecture. This includes IBM's z800, z890, z900 (with the proper microcode), z990, z9 Business Class (BC), and z9 Enterprise Class (EC). You will also need to ensure you have adequate real storage to support z/OS, DB2, and any other applications required to run in the same environment.

Operating System Prerequisites

Discussing the operating system requirements takes a little more time and is getting a little more involved. When DB2 Version 8 first became available, the published minimum operating system requirement was z/OS 1.3 (5694-A01) or above. However, one of the typical rules is to stay in school and to stay in service. Today you don't really want to be running 1.3 because it went out of service back in March 2005. z/OS 1.4 and z/OS 1.5 went out of service date in March 2007, A minimum release level of z/OS 1.5 is required for certain functions delivered in V8. Don't consider a release that is not current. Currency is always good and there may be other operating system considerations you may want to think about. For example, are you thinking about taking advantage of the System z (was zSeries in the past) Application Assist Processor (zAAP) or System z9 Integrated Information Processor (zIIP) specialty engines? If so, z/OS 1.6 (and above) is the minimum operating system required to use them. The zAAP specialty engine is for JVM workload and the (zIIP) specialty engine is used by DB2 V8. There will be a detailed discussion of the zIIP towards the end of this paper in the section "What's your incentive?". When deciding on 1.6 you have to keep in mind that it is tentatively scheduled to go out of service in the fall of 2007. z/OS 1.6 goes out of service in September 2007, so that is not current for long. So, if you have the luxury of picking which operating system you should install DB2 V8 under and want to be as current as possible, you should take a good look at z/OS 1.7 or 1.8.

Migration/Fallback APAR

Now that you have the correct hardware and operating system, you need to examine where you are migrating from. The only supported migration path to DB2 for z/OS Version 8 is from DB2 for OS/390 & z/OS Version 7. There is no skipping of DB2 versions allowed this time. You will also want to make sure your Version 7 DB2 is reasonably up to date on maintenance. The required migration and fallback support is built into the V7 maintenance stream. The V7 PTF for APAR PQ48486 (PTF UQ81009), the DB2 V8 fallback toleration PTF, is included as part of the Recommended Service Upgrade (RSU)¹ maintenance on RSU0403. This APAR must be applied to V7 before starting your V8 migration. Once APAR PQ48486 has been applied, the Version 7 subsystem must be restarted before any attempts are made to bring up DB2 V8. If a DB2 data sharing environment will be migrated to Version 8, this APAR must be applied to all members of a data sharing group before DB2 V8 is started on any one member of that data sharing group. This APAR is critical. It allows a subsystem that has been migrated to Version 8 Compatibility Mode to fallback to DB2 Version 7 in the event problems with V8 should occur. As of DB2 Version 8, the fallback toleration maintenance is no longer optional. It must be applied or the migration process will not work.

Suggestion: *At this point, long before actually starting any kind of software migration, consider scheduling at least one DB2 outage whether you have data sharing or not. When the fallback toleration PTF is applied, the DB2 subsystems must be restarted before V8 can be installed. In a data sharing situation, this service can be applied without ever stopping all of the subsystems. If you are a data sharing shop, you will want to schedule a complete data sharing outage once you have upgraded to new function mode (NFM) to enable the new locking protocol 2 and thus to improve your performance.*

Validate the DB2 Catalog

Next, you want to check out the DB2 catalog to see it is up to spec for a migration. Accomplishing this should not be a big deal if you have stayed current on maintenance. That's because APAR PQ84421 (PTF UQ85439) delivered with RSU 0406 makes available job DSNTIJP8 containing a set of catalog queries that can be used to identify "situations" in the catalog that need to be addressed before migrating to Version 8. These queries are actually delivered with DB2 V8 in job DSNTIJPM. However, you would have to install V8 to get to this member, defeating the purpose of checking out the catalog before upgrading. IBM delivers the queries as a PTF in Version 7 to facilitate their use. This is one job you do want to run. When migrating to Version 8, one of the most important concepts to remember is to read everything you can find from IBM about V8 and follow every instruction and suggestion. I can assure you that doing a little extra work on the front end can save you from having a lot of problems on the back end.

There are also multiple informational APARs that should be monitored throughout the life of the V8 upgrade process. APAR I113695 contains the latest migration information summarized in one place. This APAR includes required and suggested maintenance along with migration hints and tips. This APAR should be mandatory reading right through all three phases of the V8 migration process. In addition, there are two Unicode informational APARs that should also be tracked prior to and during the migration. The Unicode APARs are I113048 (Part I) and I113049 (Part II).

¹ For a complete description of Recommended Service Upgrade (RSU) maintenance process, refer to the IBM URL: www.ibm.com/servers/eserver/zseries/zos/servicetst/mission.html.

If V8 is in your future, and it really should be, there are still more activities you can start doing today to prepare for your inevitable migration. What follows cannot be all encompassing, of course. There are (and will be) other things you can do while waiting on the tapes to arrive.

DB2 Tools Inventory

This next short section will discuss something that could be time consuming and maybe not quite as straightforward as you might hope; doing an inventory of the tools you use to support your DB2. You can never start too soon making that never ending list of products you have lying around the shop to support your DB2 environments. You might have utilities, performance monitors and reporting tools, administrative tools, recovery tools, query tools, log tools, distributed gateway tools, debugging tools, change management and migration tools, and who knows what else or how many others. Check every department, talk with every programmer and DBA, talk with the systems folk, the performance force, security team, and anyone else you can think of that might have the slightest reason for working with or on DB2. Once you have built that list of tools, and get over the shock of what you actually have lying around your shop, start contacting the vendors that sold them to you. You will need to verify that any tool you plan on keeping around and using with Version 8 will actually work with Version 8. Remember all that stuff about 64 bit, long names, Unicode, etc... may be potential problems for some of the DB2 support products you have installed. Be especially weary of products you have had lying around for way too many years or products that are out of service and you still have individuals who insist on using them. Contact your vendors and find out what you have to do to prepare their products for that Version 8 migration. Of course, if you are using any of the IBM tools that support DB2 for z/OS (and here comes the shameless plug), they are already Version 8 enabled, but you probably already knew that. And just in case you didn't already know that and would like to learn more about them, check out the IBM DB2/IMS Tools Web Site:

www.ibm.com/software/data/db2imstools/

Running the IVP

Another pre-migration task you should consider is preparing to run the Version 7 IVP. Yup, you're reading that sentence correctly. I know you're thinking that the IVP step is part of the V8 installation process. Well, that's only kind of right. There is a set of sample databases and IVP jobs laid down during the Version 8 migration and implementation process. However, they are designed to validate the DB2 Version 8 installation/migration after you complete the move to new function mode (NFM), not compatibility mode (CM). Most of the really cool stuff you get with V8 isn't available until NFM. (*Notice the word "most" in that sentence. We will discuss what is and in not included in each mode at a later time. For now, let's get back to the discussion on IVPs.*) So trying to verify a successful migration to compatibility mode would seem to be just plain silly using a bunch of jobs that rely on being at NFM. Do you just not check out your migration to V8 CM? I could never recommend not testing. If you want to check that your migration to CM worked as planned, you will want to run some kind of IVP. However, in this case, you need the sample databases and IVPs from Version 7. Yes, you are reading that correctly; you need the DB2 Version 7 sample database and IVP jobs. If you no longer have them sitting out on disk just on the off chance that you might want to use them

again sometime, you are going to have to rerun the portions of the Version 7 installation CLIST to reinstall them.

DB2 Catalog Cleanup and Reorganization

OK, so the inventory has been completed, vendors have been contacted, and you have verified that the V7 sample databases and IVP jobs still exist or they have been re-installed, but your V8 tapes still haven't arrived (or maybe haven't even been ordered). So what is one to do if one wants to continue preparing for the arrival of Version 8? You could start to clean up the DB2 catalog. If your shop has been using DB2 for a while, you may have accumulated all kinds of stuff (objects) in the catalog that you no longer use, need, or even want. Anything you can delete from any of the catalog tables becomes one less thing that has to be converted later on (remember CATMAINT still exists and now even comes in a second flavor). Once you have sufficiently cleansed the DB2 catalog, reorganizing the catalog table spaces, or at least the ones that you are allowed to reorganize in Version 7, would seem like an appropriate next step. This is not only a good practice for the day that you migrate to enabling new function mode (ENFM), it may also help out with catalog performance, and the elapsed time, of the ENFM migration step. (By the way, the discussion of the three phases of the Version 8 upgrade (CM, ENFM, and NFM) is still coming up.) ENFM is the V8 migration mode that converts the DB2 catalog, or at least most of the catalog, to Unicode and long names. That's right, EBCDIC is out, and Unicode is in. The REORG utility takes on the responsibility for the conversion of the catalog table spaces from EBCDIC to Unicode. Having the catalog well organized before making the move to ENFM could improve the performance of the REORG utility. The REORG utility with SHRLEVEL REFERENCE also becomes available for all DB2 catalog table spaces during CM.

You could also have a potential problem with the directory table space SPT01 during either CM or NFM. This will be discussed in detail a little later.

Stored Procedures

Let's see what else you might be able to clean up before starting the V8 migration. Well, on the off chance you are still using DB2 managed stored procedures, now would be a great time to start their migration to Workload Manager (WLM) managed stored procedures. It's a fact that you will like stored procedures a whole lot more when you run them under WLM. There are lots of cool things you can do with them if they are WLM managed, not to mention that DB2 managed stored procedures are not supported in Version 8. So fire up RRS; set up those WLM service classes; add WLM ENVIRONMENT names to your procedures with the ALTER PROCEDURE SQL statement, and begin testing. Not only could your stored procedures realize better performance, you will be able to take advantage of a few procedure keywords that are only available when using WLM.

DB2 managed stored procedures will continue to function in Version 8. However, you will be unable to use an ALTER PROCEDURE against them. If you attempt to ALTER an existing stored procedure after upgrading to V8 CM, the procedure will be converted to WLM managed. Of course, you will also only be able to CREATE new stored procedures as WLM managed. Migrating all of your DB2 managed procedures to WLM managed while still in Version 7 will make the process easier and more under your control.

Plans/Packages

What about all of those good (and old) plans you have lying around? If you have plans that have not been bound since George Sr. was president (or worse yet, since Reagan's days), then you have another issue you should start planning to address (no pun intended). It has always been an age old problem whether to bind a plan (and now packages) when migrating to a new version of DB2. Some would rather never do a bind for fear of introducing new access path problems into their applications, while some do binds faithfully in an attempt to avoid access path problems and see they are always using the best the optimizer is capable of giving them. Both strategies have merit, although I favor the later. I believe you are usually going to come out ahead in the game if you do the bind rather than avoiding the bind. If performance issues arise because of an access path change, open a PMR (ETR) and let DB2 Level 2 support solve it for you. Adding trick SQL to an application or completely avoiding the bind process simply delays the inevitable. Without getting into a deep discussion on the philosophy of doing a bind, for now let's just say that if you have a DBRM or plan created prior to DB2 V2.3, it is not going to work once you get to DB2 Version 8. If you have migrated to packages, a feature delivered in DB2 V2.3, chances are you are OK. Still, if you have plans and packages that have not been bound since V2.3, you could still have issues. They may not be issues that prevent something from working successfully in Version 8, but issues just the same. Just think of the number of optimization enhancements and optimizer changes you have missed out on in the last 13+ years. In fact, if you haven't bound a plan or package since DB2 Version 5, you're probably missing out on more optimization enhancements than I have time (or space) to discuss here. You may want to consider putting a plan in place now, **before** starting your migration to Version 8 compatibility mode, which would allow you to bind your packages (and plans if those plans have DBRMs bound directly into them). Make sure you retain all of the EXPLAIN and accounting trace output for comparison between packages bound in V7 and package bound in V8. You also want to retain output from V7 or V8 compatibility mode (CM) binds for comparison to binds that happen in new function mode (NFM). That EXPLAIN and accounting data is your tool to determine changes in your package's access path. There are other tools available (like Path Checker) that will also assist you in determining where access path changes have occurred.

By the way, if you are using packages (as well you should be by now), I keep referring to not having bound a plan in a long time. I want to remind you that I am only referring to plans that have had DBRMs bound directly into them. If your plans are using a PKLIST, and accessing only packages the way they should be, then you only need to be concerned with binding those older packages.

Remember also that it's never too soon to start pumping everyone up in anticipation of the eventual migration to DB2 for z/OS Version 8, the most significant DB2 yet. Start having "brown bag" sessions, classes, and meetings to start discussing the functionality being delivered in Version 8 and how it might affect the different organizations at your shop. You might also consider bringing in the free DB2 Migration Workshop put on by IBM. If you run out of planning task, or just need a break, check out Roger Millers article (available from the web) "Migration to DB2 UDB for z/OS Version 8: Greatest Hits and Myths". Just do search on the DB2 Support web page

www.ibm.com/software/data/db2/zos/support.html

for this title or on Roger Miller to locate it. You might like his presentation on planning migration,
<ftp://ftp.software.ibm.com/software/data/db2zos/SHAREdb2zv9MigrationMiller.pdf>

The Question of Load Libraries

If you have been reading up about DB2 Version 8, you may have come across a new acronym for DB2 folk: PDSE. It stands for “partitioned data set extended” and it has actually been around for quite a while. Maybe not as long as the other acronym that we are a whole lot more familiar with: PDS or partitioned data set.

I first came across it when doing some planning on one of my first Version 8 expeditions a few years ago. In the DSNALLOC installation job step description in the DB2 V8 Installation Guide (GC18-7418-04 as of February 2006) it states (and still states in the most recent edition) that that PDSE is required for DB2's SDSNLOAD target library. This is the default for this job. This is not true; at least it's not true yet. In V8, way back when delivery was first being planned, we thought that the RDS load module might exceed the 16MB maximum size limit. That would have required the use of PDSE. However, as it turned out, nothing in Version 8 exceeded 16MB. So, as far as V8 goes, there is no need to use a PDSE for distributed load modules. This restriction is enforced for DB2 9 though. We already know we will not be able to contain all load modules to less than 16MB and PDSE is a V9 requirement. So if you are completely unfamiliar with them, it may not be a bad idea to stop by your z/OS system folk and have a discussion. Using a PDSE is not bad, it's just different. Many DB2 V8 shops have made the PDSE choice. Regardless of which you chose to use, PDS or PDSE, make sure SMP/E knows which one you picked. Be sure that PDSE service is current. There is more information about PDSE in the DB2 Installation Guide and Program Directory (GI10-8566) available from the DB2 library on the web

www.ibm.com/software/data/db2/zos/v8books.html

There is another reason you should consider the use of PDSEs. You may want to consider using PDSE for managing your stored procedures in your stored procedure address space. A PDSE allows the extent information to be dynamically updated. This could remove the requirement to stop and start the stored procedure address space when the load library goes in extents because of additions and replacements.

If you do decide to use PDSE for stored procedures or target load libraries, do some homework first. You have to take a few special precautions if a PDSE is going to be shared outside of a sysplex or GRS (Global Resource Serialization) ring. Your systems folk can explain this all. It's not a problem, just something you have to plan for before using them.

In summary: You don't need them yet, there is no problem using them (they have been around for years), and it's your choice in V8 (although they will be required in DB2 9).

What About Compilers

One area that needs to be checked out as soon as planning for a Version 8 migration begins is the language you use for programming with DB2. Plan to run with current

compilers with a new version of DB2. DB2 no longer supports quite a few outdated and unsupported programming language compilers. For an example, let's take a look at COBOL. The Enterprise COBOL for z/OS & OS/390 Version 4 compiler (5655-G53) is supported by DB2 V8 and should be the compiler you are using. However, DB2 V8 does work with IBM COBOL for OS/390 & VM Version 2 Release 2. Please note that V2R2 ended service support December 2004. Support has been removed for all other versions (out of service versions) of the COBOL compiler. If you are interested in taking advantage of the integrated SQL coprocessor, then you are going to have to use Enterprise COBOL V3.4, V3.3 or V3.2 (5655-G53) with APAR PQ83744. Please note V3R2 ended service support October 2005 and V3R3 ends service April 2007.

The following table summarizes all of the COBOL compilers and when support ended for many of them.

COBOL Compiler		Withdrawn from Service	Run-Time Supported
OS/VS COBOL	5740-CB1	Jun 1994	Language Environment (LE) Only*
COBOL/370	5688-197	Sep1997	Language Environment (LE) Only*
VS COBOL II	5668-958	Mar 2001	Language Environment (LE) Only*
COBOL for MVS & VM Ver 1 Rel 2	5688-197	Dec 2001	Language Environment (LE) Only*
COBOL for OS/390 & VM Ver 2	5648-A25	Dec 2004 (MVS only)	Yes
Enterprise COBOL for z/OS V3R1	5655-G53	Apr 2004	Yes
Enterprise COBOL for z/OS V3R2	5655-G53	Oct 2005	Yes
Enterprise COBOL for z/OS V3R3	5655-G53	April 2007	Yes
Enterprise COBOL for z/OS V3R4	5655-G53	Current Version	Yes

*Language Environment for z/OS

Although many of the compilers are out of service, older COBOL load libraries can still be supported and used with DB2 Version 8 if Language Environment (LE) is utilized. Which compilers require LE support are listed in the above table. Only the ability to precompile a COBOL program using the older compilers is a challenge. Some of that challenge, and one of the reasons for using the newer COBOL compilers, is due to the use of new function by DB2 Version 8. It is important to restate that if there is no need to use new function being delivered by DB2 Version 8, there is no need to recompile (and therefore precompile) any old COBOL programs. Their existing load modules will continue to work with Version 8 with LE.

There are similar release requirements for other programming languages. If you are a PL/I shop and plan on taking advantage of any of the new functionality in DB2 Version 8, you want to make sure you are using either IBM's Enterprise PL/I for z/OS V3.4 or above (5655-H31). IBM's PL/I for MVS & VM V1.1 (5688-235) is very old, and end of service has been announced, but it works if you avoid new function. If you will be using the DB2 precompiler services, you will require the DB2 Coprocessor provided with Enterprise PL/I for z/OS V3.2 with APAR PQ84513 applied, or a later release of Enterprise PL/I for z/OS and OS/390. V3.2 is out of service. V3.3 goes out of service September 2007. Of course, just like the COBOL compilers, there are caveats. IBM's PL/I for MVS & VM V1.1 (5688-235) compiler, although still in service, is extremely old and you should be making plans to move to a more current PL/I compiler. Version 2 and Version 3 Release 1 of IBM's Enterprise PL/I for z/OS and OS/390 (5655-H31) compiler are no longer in service. For the programming languages C or C++ both C/C ++ optional

feature of z/OS (with or without the Debug Tool) and SAA AD/Cycle C/370 Compiler V1.2 (5688-216) are supported by DB2 for z/OS Version 8.

There are available from the web migration guides for both PL/I ([SC26-3118 @ publibfp.boulder.ibm.com/epubs/pdf/ibm3m101.pdf](http://publibfp.boulder.ibm.com/epubs/pdf/ibm3m101.pdf)) and COBOL ([GC27-1409 @ publibfp.boulder.ibm.com/epubs/pdf/igy3mg10.pdf](http://publibfp.boulder.ibm.com/epubs/pdf/igy3mg10.pdf)). These should be your primary resources if additional details are needed to migrate to the more current versions of these compilers.

As with all migrations, always reference the most current version of the DB2 Program Directory for the latest and most accurate information about program dependencies. The DB2 UDB for z/OS V8 Program Directory can be downloaded at the IBM DB2 library website,

www.ibm.com/software/data/db2/zos/v8books.html.

Checking your compiler levels may not be all that is needed to support your application groups. What about all of those development tools they use with those compilers. As mentioned earlier, you want to take an inventory of all tools used to develop code that will run with DB2 Version 8 and check that they are the correct release levels for the compilers that V8 require and, if they interface directly with DB2 in any way, make sure they can handle the new functionality in DB2 Version 8.

Create a Project Plan and Build a Schedule

Version 8 affects on DSNZPARM

It is time to discuss how DB2 Version 8 affects the values you have coded in DSNZPARM. DB2 for z/OS Version 8 has made a significant number of changes (in my opinion) to the DSNZPARM macros. Not only have quite a few V7 keywords changed their default values in Version 8, some of the ZPARM keywords that existed in Version 7 have been removed from Version 8 altogether. In addition, a number of new keywords have been added to the ZPARM macros. However, there is no need to really discuss the new stuff until you at least have Compat Mode running. For now, because we are in the planning stages for our eventual migration, let's take a look at a few of the ZPARM keywords that may have an effect on your use of DB2 while you have some time to determine if further action will be required.

First, there are keywords that have changed their default. If you haven't coded a value for one of these keywords, then things could be a little different in V8 once DSNZPARM is assembled and linked; different from how they behave today in Version 7. Additionally, by reviewing the new default values being delivered in V8, you may get some hints or ideas for changes you can make to ZPARM today in V7 in preparation for Version 8. *By the way, I am not suggesting that you change a keyword simply because the default changed. I am merely mentioning the change because the default may have changed for a reason worth further investigation.* Some of the defaults changed because DB2 will perform much better with updated values.

Altogether, the defaults have been changed for at least 18 keywords; some of those changes are fairly insignificant. You can get additional information about all of the changes in the V8 Redbook (available for download from the web) "DB2 UDB for z/OS

Version 8: Everything You Ever Wanted to Know,... and More”, SG24-6079. There are a few keywords worth mentioning here though, and they may catch your attention. For example, the defaults for the DSN6SYSP keywords CTHREAD, MAXDBAT, and CONDBAT have changed from 70 to 200, 64 to 200, and 64 to 10,000, respectively. Other keywords on the DSN6SYSP macro that have had defaults changed are:

- Log apply storage: The LOGAPSTG keyword has had its default changed from 0 to 100. This is a great example of where a parameter should have always been set. If you haven't set LOGAPSTG in your V7 ZPARMs to 100 yet, you really should. You may want to take a look at my article about DSNZPARMs in the spring issue of zJournal Magazine at

www.zjournal.com

- The default for DSMAX has been increased from 3000 to 10,000.
- Checkpoint frequency: CHKFREQ keyword now defaults to 500,000 from 50,000 in V7.
- Extended security: The EXTSEC keyword on the DSN6SYSP now defaults to ON.
- Dynamic statement caching: CACHEDYN on the DSN6SPRM macro is now ON by default in V8. In addition, CACHEDYN is now modifiable using the SET SYSPARM command. Because you can change this keyword now, and DB2 cannot create something from nothing, there is always some amount of storage allocated for a dynamic cache regardless of this keyword's setting.

Then there are those DSNZPARAM keywords that are removed from DB2 once you complete your migration to DB2 Version 8. Some of them you probably never realized were there in the first place; others could present some real problems if code has to be changed once the keyword is removed from DB2 in Version 8.

PKGLDTOL could be one of the more significant keywords being excluded from Version 8. This ZPARAM keyword was delivered by APARs PQ59207 and PQ76444. With this APAR applied, a plan or package is required for the following SQL statements at the local site:

```
CONNECT, COMMIT, ROLLBACK, DESCRIBE TABLE, RELEASE, SET  
CONNECTION,  
SET :host_var = CURRENT SERVER, and VALUES CURRENT SERVER INTO  
:host_var.
```

In Version 7, PKGLDTOL could be enabled (set to YES) to circumvent this behavior allowing these SQL statements to behave as they did in Version 6. PKGLDTOL is removed in Version 8 removing the option to override DB2's behavior. It was stated in the Version 7 documentation that this keyword would be removed at the end of the V7 release.

DSN6SPRM INLISTP in Version 7 is set to 0, turned off by default. In DB2 for z/OS Version 8, INLISTP is set to 50, so is turned on by default. Because this is not on a panel, you may not have coded it. If you have not coded it, then you are currently taking the default and will take the **new** default in Version 8.

- Predicate pushdown for IN list predicates (V7 APAR PQ73454)
- Correlated subquery transformation enhancement (V7 APAR PQ73749)

A few more DSNZPARM keywords that existed in DB2 Version 7 but have been completely removed from DB2 Version 8 are OPTCCOS1, OPTCCOS2, and OPTSUBQ1; all on the DSN6SPRM macro. These three keywords were provided via APAR support in V7. Although these keywords no longer exist in V8, the functionality provided by these particular ZPARMs has been incorporated into the Version 8 base code. Minimum problems were reported in V7 by the functionality provided by these DSNZPARM parameters when they were turned on and most customers enabling these keywords reported performance improvements. In the rare case when performance degradation was detected and reported, the additional statistics available in Version 8 reversed the issue. I would strongly suggest reviewing the following APARs:

- Both PQ50462 and PQ81790, introduced OPTSUBQ1 to fix a problem with the access path selection logic to correctly add up non-correlated subquery costs
- PQ65335 introduced OPTCCOS1 to fix the problem when an inefficient access path might be generated for an SQL statement when a table is referenced by two or more predicates.
- PQ84158 introduced OPTCCOS2 to fix a problem when an inefficient access path or an inefficient index was picked for a correlated subquery

Reviewing the above APARs will give you some understanding of what these DSN6SPRM keywords will do for your V7 subsystem (or maybe to your V7 subsystem, depending on your view point) and what behavior changes will result from modifying them.

An interesting, sometimes overlooked ZPARM that existed only in Version 7 was UTLRSTRT. This keyword was introduced to DB2 by APAR PQ72337. If set to "ON", the behavior of a utility is affected by enabling automatic restart. This auto restart function is incorporated into DB2 V8, so this keyword was removed from DSNZPARM in V8. You may want to consider enabling this keyword in V7 so you can "get used to" this new behavior. If this DSNZPARM keyword is NOT enabled in Version 8, the DB2 utilities from IBM will have a different behavior than when they ran in Version 7.

Remember, the more pre-planning and clean-up you perform while still running Version 7, in preparation of upgrading to Version 8, the better off you will be.

There could be a ZPARM out there just waiting to ruin your V8 upgrade

I actually have my fingers crossed that this is one of those warnings that only affect a small handful of customers upgrading to DB2 V8. I mentioned this ZPARM briefly elsewhere in this paper. However, it was just a few sentences in the middle of a whole lot of other stuff about upgrading and I thought it should be detailed in its very own section.

As part of your V8 upgrade planning you need to check the source for all of your DSNZPARM members (development, test, QA, production, everywhere) for the **PKGLDTOL** keyword on the DSN6SPRM macro. If it is coded, make sure it is set to **NO**. If you can't find it anywhere (it hasn't been coded), then that's goodness because the default is NO. This is one parameter you do not want to have set to YES in any of your V7 subsystem.

Now that I have made a big deal out of this one single keyword, maybe I should mention what the heck it is. Remember DB2 V6? Well, back in the days of DB2 Version 6, for

whatever reason (and the reason is not really important or germane to this conversation), we (DB2 land) allowed you to get away using a few selected SQL statements in an application without the use of a plan or package. You could, in V6, get away with using this small select group of SQL statements without performing a bind. It wasn't supposed to work that way, but it did. Those statements were:

- CONNECT
- COMMIT
- ROLLBACK
- DESCRIBE TABLE
- RELEASE
- SET CONNECTION
- SET CURRENT PACKAGESET
- SET :host-var = CURRENT PACKAGESET
- SET :host-var = CURRENT SERVER
- VALUES CURRENT PACKAGESET INTO :host-var
- VALUES CURRENT SERVER INTO :host-var

How does this become a problem in DB2 Version 7? It was "fixed". The assumption was made that no one would ever try doing something like this. Surprise, customers knew about the loophole and they did take advantage of it. For V7 only, SVL delivered a DSNZPARM keyword that would allow those customers that did take advantage of the "feature", a way of successfully migrating to V7 and time to put the packages necessary in place. A new ZPARM, PKGLDTOL, could be enabled (set to YES) to circumvent this new V7 behavior and allowing these SQL statements to behave as they did in Version 6. PKGLDTOL is removed in Version 8 (gone, eliminated, no longer exists), thus eliminating any option to override DB2's behavior. Future removal of this keyword was documented in the Version 7 manuals. Here is a quote from the DB2 Installation Guide (GC26-9936-04): "You should use PKGLDTOL only to assist you in migrating your applications to DB2 Version 7. In future releases of DB2, a package or plan will be required at the requester." This meant the loophole would be permanently closed in the next release of DB2. And guess what? That next release (in our case Version 8) and the next one, DB2 9, have arrived.

Do yourself a favor, and verify that PKGLDTOL is not turned on in your subsystems. If it is, then your next step is to determine if any of the above mentioned SQL is being referenced in an application without the use of a plan or package. Once you find the applications, get them fixed so you will not have any hidden inhibitors to a successful V8 migration.

Suggestion: *If you would like to read more about this keyword, check out Chapter 7, the section labeled "Installation Step 5: Define DB2 Initialization Parameters" in the DB2 Installation Guide (GC26-9936-04). Make sure you are checking in the most recent version of the manual that is available. You can also lookup the APARs PQ59207 and PQ76444 using the IBM DB2 Support webpage..*

There is another keyword that slipped into Version 7 that you should know about. Of course, this one will not cause you any issues if you ignore it. However, it is behavior that will be built into Version 8 and the ZPARM will be removed. You can start taking advantage of it now in Version 7 and become accustomed to its behavior and avoid any potential surprises later on.

You would think that after presenting on DSNZPARM at so many conferences over the last few years that I would be up on what is changing. Not so. I am just like you all, the customers. I either accidentally stumble upon a parameter change, someone from the lab tells me about it, or some other odd way.

Well this time, it was the “some other odd way”. I was sitting through Bryan Smith’s presentation on RUNSTATS, when he mentioned a ZPARM that I had never heard of before; and it sounded interesting. So, it was off to research this new DSNZPARM keyword. The APAR detail is available on the web: PQ96628: ONLINE REORG CLAIM / DRAIN DEADLOCK AVOIDANCE.

The APAR that delivered this ZPARM came about because of a utility deadlock situation. Remember the old adage, “Do as I say and not as I do”. It fits here. We tell customers to be careful of the order when they access multiple objects to avoid getting into a deadlock situation. That is what we didn’t do. The problem was, before this APAR, SQL jobs would get claims on objects in no particular order, index before the table space or table space before the index. It just wasn’t concerned which occurred first. The ZPARM switch added in this APAR, if set on, will cause DB2 to always acquire claims on table spaces (and partitions) before acquiring any claims on any indexes. It does the same thing during utility processing. For those utilities that have to get drains on multiple objects, drains will be taken on table spaces (on the data) before attempting to get any drains on the indexes.

The ZPARM, CLAIMTBA, is a keyword on the DSN6SPRM macro. It has two settings, on (YES) or off (NO). By default, it is set to NO in DB2 Version 7 so current behavior remains the same until you (the customer) take some action. Setting CLAIMTBA to yes forces the above new behavior. This DSNZPARM keyword does not exist in DB2 Version 8. V8 accesses data first, and then the indexes.

If you decide that this APAR is of interest to you, the PTF is part of RSU 0506. You should also apply the following two APARs along with the original one mentioned above:

- PK07739: INCORROUT FOR MESSAGE DSNT397I DURING DISPLAY DATABASE
- PK09781: LOOP IN DSNB1DCM DUE TO A BROKEN PBA CHAIN 05/08/11 PTF PECHANGE

Planning Wrap-up

If you read my bio, you would have noticed that I was a systems programmer for quite a while before joining IBM. (*Yes, I did actually have a real job for about 10 years.*) Maybe because of that background, I am real big on checking on stuff in an attempt to avoid problems rather than waiting for a problem to occur and having to figure out how to fix it. So, along those lines, I think it is critical that at some point prior to beginning your Version 8 migration, you need to do a health check of your DB2 environment. There are lots of organizations both internal and external to IBM that can do health checks for you. Or you could just go out to the IBM DB2 support website, and download John Campbell’s health check presentations that he has given on numerous occasions at various conferences around the world (IDUG, SHARE, DB2 Tech Conference to mention just a few). John’s presentation, in two parts, is available on DB2’s Support webpage. Just do a search on G11jc.pdf (part 1) and G12jc.pdf (part 2) or “DB2 Subsystem Health Check”. Although both papers make for a pretty good read and the basis for building

your own health check plan, nothing compares to hearing John present them in person if you should have the occasion.

Earlier I discussed reference materials. When you are doing your migration planning, make sure you have the most current copies of the documentation needed to perform the migration for reference. Take the time to download a new, up to date, set of manuals from the DB2 web site

www.ibm.com/software/data/db2/zos/v8books.html.

You also want to make sure you have access to the right books for the task you need to perform. If you are a data sharing shop, part of the migration to Version 8 is in the data sharing manual. If you use RACF, then you will need the RACF Access Control Module Guide. If you use Java, then you need to have Java manuals for that portion of the migration/installation. While deciding what manuals you might need to complete your planning, don't forget about z/OS. It never hurts to have access to a set of current z/OS manuals to answer those occasional questions that might come up. While I am mentioning z/OS, check out that Unicode service is installed and installed correctly, has all of the required PTFs, and is ready for DB2. DB2 Version 8 will not even start if Unicode services are not configured correctly.

I have also already talked about maintenance. Hopefully, you are well on your way to validating that all required DB2 Version 7 maintenance has been applied. That must be completed before you start your migration. While you are checking on maintenance, also make sure you have adequate disk space for the SMP/E Version 8 installation steps.

To close out this entry, I want to emphasize that the three most important things to remember about migrating to Version 8 are: get some education, read the documentation (and I mean really read it, don't assume anything from previous migrations), and take it all very slowly, one step at a time, checking everything you complete. It may be a trite expression but it applies so well to a Version 8 migration; "*The faster you go, the further behind you get*". So do not short the planning steps.

DB2 for z/OS Version 8 Utilities use of DFSORT

Before going too far down the compatibility mode road, some time should be spent discussing DFSORT. If you use the IBM DB2 utilities and any non-IBM sort product, this is important for you to understand before using the IBM DB2 utilities. If you do not use the IBM DB2 utilities or DFSORT is your sort product, you can skip over this section.

In the last week, I have again had a number of inquiries about DB2 for z/OS Version 8 and its use of DFSORT. This was a subject heavily discussed on IDUG's DB2 Database Discussion list (DB2-L listserve available at www.idugdb2-l.org/archives/db2-l.html) a while back and a subject I thought had been clearly explained. Well, it is once again a topic of discussion on DB2-L. So with the resurgence of interest in DFSORT, I thought I would add a few sentences here with my opinions.

First and foremost, **the IBM DB2 for z/OS Version 8 utilities require DFSORT**. This support was delivered by APAR PQ68263 (PTF UQ90054) on RSU0312. In fact, with the delivery of APAR PK04076 (UK03983), MSGDSNU1640I will be issued by DB2 when the IBM DB2 utilities are unable to locate DFSORT code at APAR PQ68263

maintenance level. So DFSORT is mandatory and you have to use it for the utilities. However, there is good news.

You already have DFSORT installed. You may not be using it, but it is there. DFSORT is always shipped and installed with z/OS. However, no license is required for the DB2 utilities to use DFSORT. The only action a customer must take is to add the DFSORT load libraries to link list after the OEM sort libraries or add a STEPLIB/JOBLIB to the DB2 utilities batch JCL. **DB2 for z/OS Version 8 has a license to use DFSORT** so the customer does not need a license for any external sort product. Again, MSGDSNU1640I will be issued when the utility is unable to locate DFSORT code. However, no non-DB2 task can use DFSORT without being licensed. Use of DFSORT other than by DB2 would require a license.

Maintenance for DFSORT is shipped with z/OS. If a customer has problems with DB2's use of DFSORT, they would open an ETR with DB2 Level 2 Support no different that they would do today for any other DB2 related issue.

The DB2 Version 8 requirement for DFSORT R14 or above is described in the following publications:

DB2 Version 8 Installation Guide (GC18-7418) (All V8 manuals can be found at www.ibm.com/software/data/db2/zos/v8books.html)

DB2 UDB for z/OS Version 8: Everything You Wanted to Know... and More (SG24-6079)

Always refer to the latest updates of the manuals from the appropriate IBM web site before making decisions.

The DB2 Version 8 utilities use DFSORT and only DFSORT. You have no other choice. There was not a whole lot published on the subject before the APAR. And a few customers thought this was just a suggestion because some utilities seemed to work just fine with their non-DFSORT sort product. Then some customers started to get a strange message that the utility they were running could not find certain DFSORT modules. It is then that everything seemed to hit the proverbial fan. As a result of all of the questions that started to pop up, an APAR, PK04076 (PTF UK03983), was published enabling a new error message. MSGDSNU1640I is now issued when a utility is unable to locate the DFSORT code after applying the APAR PQ68263, the maintenance that enabled the exclusive use of DFSORT by the V8 utilities.

Having DB2 for z/OS use DFSORT allows DB2 utilities to test and support DB2 more thoroughly. In V8, the utilities allow sorting for 32K pages, which works with DFSORT.

Now, having the utilities take advantage of DFSORT, and ONLY DFSORT, is a good thing. In fact, **it's a great thing**. DFSORT is one of our products. We know everything there is to know about DFSORT (well in theory at least) and at the very minimum we should know more about our DFSORT than other vendor's sort products in use today. So, by making the DB2 V8 utilities use DFSORT exclusively, we have the opportunity to take advantage of functions and features in DFSORT that we have not been able to consider in the past. Why? Because now, we do not have to worry about a utility using some sort feature/function that we have to check will work with every sort product on earth, not just DFSORT. That should mean that when the dust clears, our DB2 customers win. *Of course, this is just my opinion about why DB2 now uses DFSORT exclusively, not something I was told, overheard, or read in a book someplace.* So

please don't tell your IBM sales rep, "Willie said so". It will not get you anywhere. But let's get back to some useful information.

So what are some of the questions I am getting and more importantly, what are the answers.

The big question is always "I'm not licensed for DFSORT so how can I use it?" Although it is true that you are not licensed to use DFSORT, the DB2 utilities are. They have a special license to use the necessary functionality they need in DFSORT. Of course, you cannot use DFSORT outside of the DB2 utilities because as you stated, you are not licensed.

Then someone will ask if all the utilities use DFSORT. The answer is No. Of the DB2 utilities, only LOAD, REORG TABLESPACE, REORG INDEX, REBUILD INDEX, RUNSTATS, CHECK DATA, CHECK INDEX, and CHECK LOB use DFSORT.

One of my favorite questions is "How does DB2 know to use DFSORT and not my *other* sort product?" This question has a few answers. It is the responsibility of the customer to make sure that the load libraries are in the right place and in the correct sequence. The two DFSORT libraries have to be in the search list **AFTER** the sort products libraries in use. So for example, if you are using SyncSort and running out of LINKLIST, make sure the SyncSort libraries are first, followed by the DFSORT libraries. By the way, this is the configuration I see the most and the one that I prefer. You can also STEPLIB or JOBLIB to the DFSORT libraries but that seems like a lot of extra work and requires you to modify existing JCL and PROCs. The other half of this answer is that the DFSORT load modules called by the DB2 utilities have different names than the usual sort modules that are invoked. So unless a vendor is going to start using the same names IBM uses, you should not have a problem. In either case, the IBM modules will always be second in the search order if the libraries are added correctly.

So what are these special names DB2 uses you might ask? Although there is no reason for you to ever have to use or find them, the DFSORT module aliases called by the DB2 utilities are ICEDFSRT and ICEDFSRB. The module normally called when invoking sort (anyone's sort) are ICEMAN and ICEBLDX.

Now the next question, when not asked, seems to cause the most headaches. "Do the DFSORT load libraries have to be authorized?" Absolutely! DB2 runs authorized, so the sort libraries must be authorized. If you are going to use the LINKLIST idea, you can authorize them by specifying the keyword LNKAUTH=LNKLST. If you use LNKAUTH=APFTAB, then you will have to add each of the libraries to the APF authorization list. If you do not authorize the DFSORT load libraries and you attempt to run a DB2 utility, the utility job will fail.

Over the past few months I have made a number of posts pointing you to all kinds of DB2 documentation resources. While I was preparing my first Compat Mode entry, I remembered that **IF** you are doing planning for DB2 Version 8 or if you are currently testing in V8 Compatibility Mode, I should warn you about the Version 8 documentation. All of the product documentation makes that assumption that you are in New Function Mode (NFM). If something in the manuals refers to a feature/function in Compatibility Mode, it will normally explicitly point that out. However, in general, you should always

assume that what you are reading about in the DB2 Version 8 product manuals is talking about something in NFM.

DB2 Version 8 DFSORT informational APARs

Before you get too far into this, make sure you check out the three informational APARs that describe how the DB2 utilities in Version 8 use DFSORT. When you locate it on the web, bookmark it, and then visit it often or least make it your first stop when you have questions or issues about what DB2 is doing with its relationship with DFSORT. This is where DB2 is currently documenting the latest information about how DFSORT and DB2 work together. The informational APARs (II14047, II14213, and II13495) are available from the IBM web site at

www.ibm.com/support/docview.wss?rs=0&uid=isg1II14047 (last update 10/2006)
www.ibm.com/support/docview.wss?uid=isg1II14213 (last update 10/2006)
www.ibm.com/support/docview.wss?uid=isg1II13495 (last update 08/2006)

When you do find the above APARs on the web, make sure the "Last Changed Date" is at least the date listed after the URL. These are the most current versions of the info APARs as of the date on the cover page of this paper.

DB2 for z/OS Version 8 Reserved Words

Yup, every new version and release of DB2 has them. They are words that you cannot use in an SQL statement under penalty of error (-199 or DSNH083I) unless they are a delimited identifier (marked off by a quotation mark (") for example). And why might they be a problem? Well, DB2 might just think they are DB2 keywords and get all confused. The situation is a little more complicated, since DB2 can use many keywords as a name or identifier, if DB2 can tell from the context of the statement that it's not a keyword. So when you move to V8, you might discover that you have been using a reserved word from V6, but the context has not caused a problem until now.

So, to avoid that problem, don't use them. In fact, there is an entire table of DB2 reserved words in Appendix B of the SQL Reference manual (SC18-7426). And make sure you have the most current edition of the manual (I used the -03 from Feb 2006 for this paper). I know I keep repeating this throughout this paper because it is important. You don't want to go through all of the trouble of making a list of reserved words that should be avoided only to discover you missed one because your documentation was out of date.

Although I am telling you to always check the latest SQL Reference, I am posting a list of reserved words added in DB2 V8. The list should be current as February 2006. Check this list carefully if you are running V7 and planning an upgrade to V8. This list does contain a few words that could trip you up. If you want to be more careful, check the list of reserved words in the SQL Reference for Cross-Platform Development, which includes words for the DB2 family and also the list of standard reserved words.

www.ibm.com/developerworks/db2/library/techarticle/0206sqlref/0206sqlref.html

ASENSITIVE
ENCRYPTION
ENDING
EXCEPTION

HOLD
INCLUSIVE
ITERATE
MAINTAINED

MATERIALIZED
NEXTVAL
NONE
PADDED

PARTITION	REFRESH	SIGNAL
PARTITIONED	RESIGNAL	SUMMARY
PARTITIONING	ROWSET	VALUE
PATH PIECESIZE	SECQTY	VARIABLE
PREVVAL	SECURITY	VOLATILE
QUERY	SEQUENCE	XMLEMENT

SYSOPR

*"Oh what a tangled web we weave..."*²

And DB2 does, if you are not careful. There are so many things that affect other things in DB2 that we sometimes take for granted.

Take something as simple as entering a DB2 command at an MVS console (if you're logged on) or through SDSF. For a lot of people, it's something that just happens. And it does "just happen" because DB2 assumes the authorization identifier (authid) with that kind of request is "SYSOPR". No, not the DB2 privilege set SYSOPR, this SYSOPR it's just another authid. So why did the command work? This is where the web gets tangled. When you install DB2, the install SYSOPR authid in DSNZPARM by default is set to SYSOPR. Yup, I kid you not. And if you don't change it, that is what it remains. Would you like to know how many DSNZPARM source members I have examined over the years as part of a DB2 health check that were still using the default? Let's just say more than you would believe. Of course, as you already know, that lets you enter the command.

Here comes the classic good news, bad news joke.

The good news side of this story is that everything that I just described is DB2 Version 7 behavior. The possibly bad news side: it changes in Version 8. Actually, if you are security conscious and concerned about who can run your DB2 commands like '-STOP DB2' then this is really all good news.

In DB2 Version 8, the USERID signed onto the console or using SDSF is now checked to see if that id has the correct set of privileges to issue a DB2 command. If you didn't mess around with passing out this authority in V7 and just used the ZPARM default, you are going to have a lot less DB2 commands being issued from SDSF or from the console. I would suggest straightening this all out before you upgrade to Version 8 so you don't have to worry about stuff not working once your upgrade is complete and you, as an added benefit, you will fix a possible exposure in V7.

Some added good news in V8: Authority for commands can be a secondary authid. This could make it easier to set up the right people to have this privilege now that command access has to be explicitly granted. Install SYSOPR is also modifiable using the SET SYSPARM command. The only catch to doing this is that you have to be the install SYSADM when you issue the SET command. If you are not, all other DSNZPARM changes will take affect except the change to install SYSOPR.

² Scottish novelist Sir Walter Scott

The report of “excessive CPU consumption” was an exaggeration³

In this section, I would like to spend a few minutes discussing DB2 for z/OS Version 8 CPU consumption; what's true and what is a myth. With V8, you need to remember that all you hear (and sometimes see) isn't always completely accurate. For example, what **is** true is that any new version of DB2, including V8, will probably use more CPU than the previous version. It's just the nature of the beast and the cost of doing business (business meaning improved usability, availability, scalability, and that long list of ...ability's that we always talk about). However, V8 brings some other “challenges”. Although 64 bit processing brings us the availability to do lots of cool storage stuff, it also brings us instructions and addresses that can be up to 50% larger. Larger instructions and addresses mean bigger registers and control blocks. This can all translate, in some situations, to additional CPU when the same v7 workload is run in V8. Now, how much of a CPU increase you will actually experience will vary considerably depending on what kind of workload you happen to be running. In fact, some customers have seen no (and in some cases a negative) CPU increase. Hey, go figure! It is DB2 and although it is a horribly overused expression, with DB2 it is always “it depends”.

Bind CPU cost, especially for dynamic prepares, may go up if you have started to use the RUNSTATS frequency value option and the FREQVAL COUNT in SYSCOLDIST is greater than 100 because Bind is attempting to remove dups. If this is happening to you, check out APAR PQ94147. This is an older APAR and demonstrates the importance of staying as current as possible on maintenance.

Then of course there is one of my personal favorites, multi-row FETCH. I have talked multiple times about the value of this feature. But, did you know that you don't have to wait until new function mode (NFM) and actually coding a multi-row FETCH statement in an application program to take advantage of the CPU savings of this feature. When you are in compatibility mode (CM), all you have to do is have an application use DRDA's limited block fetch and you will automatically get to use multi-row FETCH and receive immediate CPU savings.

And then there is the case where you are seeing high CPU consumption that really isn't there. You need to take a look at APAR PK25427 for DB2 and the z/OS co-APAR OA15666 that “fix” the way storage is being reported back to DB2.

You can also start to look into using long term page fix for buffer pools, selectively at first, until you understand the real storage impact. Page fixing the buffer pools can reduce your CPU usage just like eliminating I/O can decrease your CPU usage. ...and you eliminate I/O by keeping more in the pools. ...and you can keep more in the pools if you have more in a data page. ...and you can get more into a page if you turn on compression. A long way to go, I know, but it was worth it. Compression is something you should be taking advantage of in Version 8. Now that you have access to more storage, some of the storage restrictions that prevented you from fully utilizing compression have been lifted. So, get out there and start considering turning on compression for more table spaces.

³ *My attempt here was to paraphrase Mark Twain's quote “The report of my death was an exaggeration.” However, while trying to find the correct wording of Mr. Clemens' famous quote; I discovered that most “misquote” him anyway.*

Another “myth” area you need to be aware of is storage consumption. An example of a virtual storage increase that you should be aware of was caused by an increase in the number of write engines from 300 to 600. It seemed like a good idea and the added storage usage seemed worth while. But it was discovered this enhancement didn't give quite the kick that was expected especially considering the additional expense it added. So APAR PK21237 came out last March 2006 (PTF came out on May 2006) that reduced the number of write engines back down to 300. This will immediately buy you back on the average, about 100 MB of storage without any effect on your performance. This APAR will also fix a few things that have to do with releasing stack storage and combining engines to a single pool. Not bad for just one PTF.

Final comment: Virtual = Real. A lot of stuff we talk about with V8 suggests increasing pool sizes. But you have to remember, even though you are in a 64 bit environment, storage is not unlimited. You can only use what you have backed completely with real. This means 100% backed, nothing less.

Compatibility Mode (CM) Overview

You have completed all of the suggested planning, and the tapes have finally arrived or you have completed your downloads from ShopZ. It is finally time to start your migration from DB2 Version 7 to Version 8. The most important thing you want to remember now is that this is just a migration. Sure, V8 is probably the most significant new release of DB2 since Version 1. However, it is still just a migration. If you have studied the documentation, have a good plan in place, and follow all of the steps as described in the DB2 Installation Guide (GC18-7418), assuming nothing, skipping nothing, and guessing at nothing, you will be just fine. There have been thousands of successful migrations completed already.

We (that group of individuals that runs around the country presenting on DB2) have done a great job of making a lot of customers think they are doing something new and different from any previous migrations they have completed by introducing a bunch of new terms to the process. You are, kind of, and at the same time, you are not. DB2 for z/OS Version 8 introduces three distinct steps to the migration process; compatibility mode (CM), enabling new function mode (ENFM), and new function mode (NFM). But are they as new as they might appear? CM we will discuss in a minute. ENFM and NFM are steps designed to formally prevent the use of new function until you are ready.

Let's start talking about compatibility mode (aka compat mode or simply CM). CM is the first step in a Version 8 migration process. Most of the tasks that we have become accustomed to from previous version migrations occur during the CM portion of your V8 migration. This is the step where you run CATMAINT to make changes to existing tables, columns, and indexes or to add new table spaces, tables, columns, or indexes to the DB2 catalog. And although getting CATMAINT out of the way is a significant step in the migration process, maybe one of **the** most important functions of compatibility mode is your first introduction to DB2's use of 64 bit processing. The DBM1 address space has been updated to run in a 64 bit environment. When you are in CM you are insuring that your DB2 functions correctly in this environment that is new to DB2.

When we present, we often make the comment that very little new function is delivered in CM other than 64 bit. That is not entirely true. The following list are “*some*” of the DB2 for z/OS Version 8 features and functions that are made available while running in compatibility mode. This list is neither all inclusive nor is there any guarantee that this list will not change in the future as service is applied to DB2 Version 8.

You **should not** make yourselves dependent on any of the items listed below being available in compatibility mode other than basic 64-bit.

- Database Services Address Space runs in 64 bit mode. There is no dual path code available. This is why there is a z/OS 1.3 (now out of service) or above is a requirement for a V8 migration.
- IRLM V2.2 is required and must run in 64 bit mode. Although IRLM V2.2 can run in 31 bit (IMS support) and 64 bit mode, DB2 V8 requires that it run only in 64 bit mode. Lock storage is above the bar allowing an increase in the number of locks stored (NUMLKTS and NUMLKUS) assuming adequate real storage is available.
- Buffer pools are moved above the bar. Data spaces and hiperpools are eliminated
- DBD section of EDM Pool (EDMDBDC) is moved above the bar
- Dynamic statement cache section of EDM Pool (EDMSTMTC) is moved above the bar
- Sort pools are moved above the bar
- Castout buffers are moved above bar
- RID Pool is moved about the bar
- Compression dictionaries are moved above the bar
- Most optimization changes are in affect and can be taken advantage of if plans and packages are rebound.
- New access paths in plans and packages may take more space so the size of the SPT01 and SCT01 might need to be increased.
- New catalog and directory objects and new columns are added to existing catalog tables (CATMAINT is run when migrating to compatibility mode)
- DB2 parses all SQL in UNICODE
- String constants may be longer when represented in UNICODE which can result in some string constants exceeding the maximum length for a string constant
- PGFIX(YES) option on the ALTER BUFFERPOOL command is available
- IRLM PC=YES enforced. PC-NO is ignored in V8
- z/OS Conversion Services is used for CCSID conversion to UNICODE
- Stored procedures can no longer be defined or run with COMPJAVA
- Online REORG SHRLEVEL REFERENCE of entire DB2 catalog is available (and used during ENFM migration to convert catalog to Unicode). This does not mean REORG SHRLEVEL CHANGE is available.
- Larger buffer pools are immediately available (provided that you have enough real storage). You can alter buffer pools to long term page fixed.
- Mismatched data types become stage 1 (indexable) predicates in many cases.

You cannot have a discussion about compatibility mode without talking about coexistence. You're right; coexistence only applies to those customers migrating their data sharing group to Version 8.

CM Coexistence

I must be missing attending English class because this section had me searching for a dictionary to look up the definition of my title word. The current web version of the Merriam-Webster's dictionary defines coexistence as:

- 1) *to exist together or at the same time*
- 2) *to live in peace with each other especially as a matter of policy*
- 3) *the occurrence or existence of several things at once*

I just love the second definition. Wouldn't you think that all releases of a software product should "live in peace with each other"? I once tried running two different versions of a piece of software on my notebook to retain a function in the old version that I still needed. What a mess that turned into.

IBM gets more to the root of my discussion with its definition of "coexistence" in the glossary of the DB2 V8 manuals: *during migration, the period of time in which two releases exist in the same data sharing group.* You will notice IBM makes no mention of anyone getting along or any kind of "peace with each other". In fact, if you stick two different versions of DB2 in the same data sharing group, your life could get down right exciting. However, I have really jumped way ahead of myself. Let's step back and start a little closer to the beginning.

First, why did I introduce this entry with definitions of coexistence? Many customers today take advantage of DB2's data sharing. When you migrate to a new version of DB2, one of the cool things about DB2 running with data sharing is the ability to run two different versions of DB2 in the same data sharing group at the same time. It makes your migration easier (in theory) by allowing the group to stay active during the migration. The plan here is to move one member at a time to DB2 Version 8 while the rest of the group remains at DB2 Version 7 until all of the members have been migrated. Now, with Version 8, the ability to have two versions coexist is only allowed in compatibility mode.

Now there is a process that needs to be followed in order to effectively make it through all of this madness. So ask yourself these questions! First: Do you have APAR PQ48486, which can be found at URL applied to all members of your data sharing group. Once the fallback toleration APAR PQ48486 has been successfully applied, you will have to restart all of the members of your data sharing group while still on Version 7. Being able to fallback to Version 7 after migrating to V8 compatibility mode is a critical part of the Version 8 migration process and needs the same APAR applied as does running some members of a group on V7 while other members of that same group are on V8. With the application of the appropriate maintenance applied, and all of the members of your data sharing group restarted, you are now ready to start migrating members of your data sharing group from DB2 V7 up to V8 compatibility mode. Just remember, the toleration APAR must be applied and running on all members **BEFORE** any member can be migrated to V8.

Scheduling Note: *This can be a rolling outage, so that end users do not see any down time. You should schedule an appropriate time frame to incur this outage at the onset of your V8 migration to minimize the impact on your completion. The more stringent your processes are for scheduling software changes, the more important it is to take care of this at the onset of you V8 migration. While you are scheduling, you need to get the*

early or ERLY code changes on every z/OS image, and this step requires an IPL. Check for the service levels needed.

Next question: Are you using DB2 Connect and the connection concentrator option with your data sharing group? If so, you have more maintenance to check is applied. To take advantage of DB2 Connect Server connection concentrator while running in coexistence requires, at a minimum, DB2 Connect Version 8 Fixpak 10. You should be in Version 8 of DB2 Connect regardless because V7 of Connect is out of service. However, if you are going to take advantage of coexistence, you have to be on at least Fixpak 10 of DB2 Connect V8. I cannot state that too many times. You will also have to make sure you have the correct level of maintenance on DB2. Again, if you are planning to use DB2 Connect Server connection concentrator with DB2 V7 and V8 running in coexistence in a data sharing environment, you also have to apply APAR PK05198.

Before I continue, I would like to make just a few additional comments that may clear up (or possibly muddy up) some of the questions and issues about DB2 Connect. Yes, it is true that the documentation states DB2 V8 will work with DB2 Connect V7.2 with fixpak 10, 11 or 12 applied, even though DB2 Connect V7.2 went out of service back in September 2004. And yes, there was a service extension given to customers moving to DB2 V8. However, most of the really cool improvements in DB2 Connect were delivered with V8.1 and fixpak 4. But even that release level is probably not where you would like to be. If you are **NOT** planning on using connection concentrator, you want to give installing, at a minimum, DB2 Connect V8.1 with fixpak 5 some very serious consideration. If you have any plans to use DRDA encryption support, then you will need Connect V8.2 fixpak 7A to satisfy the minimum requirement, but here again you should at least be looking at fixpak 8.

DB2 Connect version and service level recap:

- DB2 Connect 8.2 fixpak 13 for current service
- DB2 Connect 8.2 fixpak 10 if you are using connection concentrator AND data sharing coexistence
- DB2 Connect 7.2 fixpak 10, 11 or 12 satisfies minimum requirement
- DB2 Connect 7.2 is out of service although an extension was given to customers moving to DB2 V8. However, the extension time has expired.
- DB2 Connect 8.1 fixpak 5 should be considered your minimum entry level (IMHO)
- DB2 Connect 8.2 fixpak 8 if you want DRDA encryption support
- Consider moving to the most current level of DB2 Connect. As of this writing, DB2 Connect V9 fixpak 2 is the most current release.

Everything you would ever want to know about DB2 Connect can be found on IBM's web site at URL:

www.ibm.com/software/data/db2/db2connect/.

You can also find all of the DB2 Connect fixpaks on IBM's web site at URL:

www.ibm.com/software/data/db2/db2connect/support.html.

Final question (for this entry): How long do you plan to stay in coexistence? I am not asking you how long you will stay in compatibility mode. You need to ask yourself how long you plan to run your data sharing group in coexistence? Your answer should be

something in terms of days, not weeks, and definitely not months. You should plan to stay in data sharing coexistence for as short a period of time as possible. Coexistence is designed to move your data sharing members to V8 while minimizing any outages. Once a member has been successfully migrated to V8, the other member's behavior should be no different than the first. They should all be running the same code. So, do everything possible to keep your time in data sharing coexistence as short as possible.

You're in V8 CM and don't like your access path. Now what?

Good question and one that I think is of concern to almost everyone when they migrate to V8 (or any other version migration). I guess you always have the option to scream "Hey DB2 land, what's wrong?" Of course, what would probably be a better approach would be to start checking out the stuff that DB2 Level 2 support might ask you to check out anyway. Do you have your EXPLAIN data? Have you checked it to see if your access path changed? Of course, just because it changed isn't necessarily a bad thing. There have been a lot of optimization changes in Version 8.

I'm sure you have already guessed where this discussion is going to head. EXPLAIN; an easy and straightforward process that could be your first line of defense against any potential access path problems. With today's DB2 though, you don't just want to be using EXPLAIN by itself. No one should have to be a PLAN_TABLE guru that has memorized all of those cute little codes written to the PLAN_TABLE by the EXPLAIN process. What you want to be using is a tool that translates all of those values for you, something like Visual Explain. If you are already using Visual Explain (VE), great! However, you do want to make sure you are using Visual Explain Version 8. Regardless of the version of DB2 you are running on the System z platform (V8, V7, heck even V6), you should absolutely be using DB2 for z/OS **Visual Explain Version 8**. The Version 8 portion of VE's name is the key to your success and the important portion of this product's name. Visual Explain can be downloaded at no charge from IBM's DB2 webpage at

www.ibm.com/software/data/db2/zos/osc/ve/.

In an earlier discussion about stuff that should be considered when preparing for your V8 upgrade, one item that almost everyone agrees on is the retention of EXPLAIN (PLAN_TABLE) information from your DB2 Version 7 subsystem. Once you migrate to V8 compatibility mode (CM), you could need that saved V7 PLAN_TABLE data to compare to your new EXPLAIN information you are going to produce when binding applications in CM.

Visual Explain is a pretty cool tool that is priced incredibly well; it's free. As I am sure you already know, the beauty of VE is its graphical representation of an SQL statement's access path. With VE V8, by taking advantage of XML, you are given the ability to save that graphical depiction, and all of the access path analysis, for future use. VE V8 has also improved the information that accompanies these graphical displays. With VE's enhanced capabilities, it now gives estimates about the number of qualified rows, more detailed predicate information (for matching, screening, Stage 1, Stage 2), improved details about partition scans and parallelism, and sort estimations. Do not assume VE V8 is a simple update of the Visual Explain that has been floating around for the last couple of years. Although it still does everything VE Version 7 did, Visual Explain

Version 8 is a complete re-write, top to bottom, of the entire tool in Java. You may be surprised at how much more useful and intuitive the V8 flavor of VE is over previous versions. For example, V8 allows you to display multiple query blocks at one time or a single query block, your choice. There are context-sensitive tuning suggestions available, you can link from the graph to additional statistics, you can catalog and uncatalog databases on your local machine, and you can use VE to run queries and view the formatted results. And you aren't dependent on just using what is displayed by the graph on the screen. You have to option in VE V8 to create an HTML report with just a click of a button.

There are a couple features in VE that have little to do with explaining an access path but are still extremely helpful. For example, let's say you are a first time user and need to create all of the tables used by EXPLAIN. Just click "Subsystem" on the tool bar and select "Enable Visual Explain" from the pull-down. After connecting to your DB2 for z/OS subsystem, all of the following tables are created:

- PLAN_TABLE,
- DSN_FUNCTION_TABLE,
- DSN_PREDICAT_TABLE,
- DSN_STRUCT_TABLE,
- DSN_PGROUP_TABLE,
- DSN_STATEMNT_TABLE,
- DSN_PTASK_TABLE,
- DSN_FILTER_TABLE,
- DSN_DETCOST_TABLE,
- DSN_SORT_TABLE,
- DSN_SORTKEY_TABLE,
- DSN_PGRANGE_TABLE,
- DSN_VIEWREF_TABLE.

From this list I am sure you notice that Visual Explain V8 uses far more tables than the old VE. All of this is there is give you the most accurate access path analysis possible.

And what if you do have an SQL access path problem? No longer do you have to try and figure out what DB2 Level 2 needs in order to work on your problem. With VE V8, from a pull down on the tool bar, you can click on "Service SQL" and have Visual Explain prepare an XML file with all of the information necessary for Level 2 to work on your SQL issue. With the click of a few keys, the XML file is created and even FTP'ed to IBM DB2 Service. It can't get much easier than that.

Of course, I cannot do justice to Visual Explain in this paper. However, because it is priced so well (it's free), just download a copy and give it a try. All you need is a copy of DB2 Connect to communicate with your DB2 for z/OS subsystem and a few stored procedures (DSNWZP, DSN8EXP, and DSNUTILS) installed to take advantage of all of the functions. If you do not have a DB2 Connect license, don't worry. DB2 for z/OS includes a license for DB2 Connect to use for this purpose. You may also want to look at APAR PQ62695 for some additional information about using the Java Universal Driver. If you need some additional information, there are lots of resources available from the IBM DB2 for z/OS Support website. Just do a search on "Visual Explain"

If you would like to take a look at a few papers on Visual Explain Version 8, check out two of Patrick Bossman's (DB2 Developer with IBM's Silicon Valley Lab) presentations "DB2 for z/OS Visual Explain V8" and "Visual Explain for V8, We Found the Beef". There is also a very good presentation containing some discussion on VE by Terry Purcell (also a DB2 Developer with IBM's Silicon Valley Lab) entitled "Major Optimization Enhancements in DB2 Version 8". All three of the presentations just mentioned are available from DB2' Support webpage by performing a search.

In addition, there are two Redbooks that also have an excellent discussion on the merits of Visual Explain Version 8:

- "DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More" (SG24-6079) - Chapter 10.19 in my copy has a good discussion on the EXPLAIN improvements in V8 and VE,
- "DB2 UDB for z/OS Version 8 Performance Topics" (SG24-6465) - Chapter 3.15, again in my copy, has a VE discussion with a great section describing how to use Stats Advisor.

The most recent version of DB2 Visual Explain Version 8 can be downloaded from the IBM DB2 website at URL

www.ibm.com/software/data/db2/zos/osc/ve/.

Compatibility Mode – How long is long enough

Migration to compatibility mode is complete, or almost complete. The next big question that comes up is "How Long?" How long is long enough for you to stay in compatibility mode? You will not want to hear the answer. It depends. It depends on how you move new releases of software into production. It depends on how extensive the testing is that you have planned in compatibility mode. It depends on whether or not you have any problems. It depends on your "window" of availability. There are so many variables that a time recommendation is almost impossible. Every time I tell a customer it should only take x amount of time, I get this long dissertation on all the things involved in migrating at their shop. Sometimes you have to wonder why they asked in the first place.

Barring the fact that you have hundreds of DB2 subsystems and data sharing groups with an extremely complex QA program for moving software into production, my answer would have to be "Stay in CM through a major event". For example, if by the completion of end of month processing, you have put the majority of your code that will use DB2 Version 8 through its paces, then EOM would be your major event. If it is end of quarter for you, then EOQ is your major event. So on and so forth. Make sure you also build all of those software migration blackout dates into your migration plan. If you are retail and do not touch code between Thanksgiving and mid January, plan accordingly. Also, do not forget about other projects. If your company has scheduled the biggest application in your history to move to production next month, next month may not be the best time to also move DB2 to ENFM and NFM.

However, the real bottom line to remember is you do not want to stay in compatibility mode for an extended length of time. It is costing you to be in compatibility mode. You are paying the price of some extra V8 stuff and not gaining any of the benefit of being on Version 8. One example is the SQL parser. DB2 parses SQL using Unicode UTF-8 (CCSID 1208). In CM your inputs may still be in EBCDIC. That means DB2 has to convert stuff from EBCDIC to Unicode and then back to EBCDIC. That is costing you. Once you get to new function mode, you are no longer incurring that cost.

So, do what you have to do. Follow all of those internal procedures and block out all of those software freeze dates. And make sure your plan includes moving to ENFM and NFM someday, hopefully sooner than later.

I will discuss ENFM in a little bit. However, because we are discussing timing and planning here, I would like to comment on how long you should consider staying in enabled new function mode. I would suggest something in days, possibly even minutes, rather than weeks or months. ENFM converts the DB2 catalog to Unicode. That is all it does. Stay there just long enough to make sure the conversion was successful, you still have a functioning catalog, and get on to the next step, new function mode. The only reason to maybe not move to NFM right away would be because you are trying to restrict the use of new features and functions in Version 8 so you can have a controlled rollout. Just don't stay there because of testing. You have tested 99% of all your going to test, with the exception of features and functions available on in new function mode, so don't spend a lot of additional time retesting what you already decided works.

And rest assured, this is a decision you will eventually have to make once you begin your migration to V8 compatibility mode. While running in compatibility mode, there could be overhead that you did not experience when you were on Version 7. In fact, there is always some degree of degradation with any version to version migration. It is also true that with every version migration that there will be performance gains. Some of those gains come from the improved optimization you could realize in V8 if a plan/package were rebound. Most of those optimization improvements are also available in compatibility mode. However, to take advantage of the optimization improvements in V8 you will have to rebind your plans and packages. It is possible that this will not be your choice in some cases. If you still have any plans running at your shop that have not been bound since before your Version 2.3 migration, those plans will not work in Version 8. It should also be a fairly straightforward task to identify those plans. Just run a query against the DB2 catalog for any plan using DBRMs directly. Almost all customers are now using (or should be using by now) packages. Packages were delivered in Version 2.3. If you are using packages, your plans will be bound using the PKLIST keyword, not with DBRMs. So, the logic here is if you have a plan with a DBRM it might be pre 2.3 and could be a candidate for rebind. You should also identify and rebind any performance critical packages to check they are using the best access path. If you are still in the migration planning stages, this would also be a good time to start gathering explain data and accounting information in your Version 7 DB2. You will not know if you have degradation in performance if you have nothing to compare to. The mere fact you have moved to a different version of DB2 will have users looking for problems. Having the ability to compare access paths and performance numbers between versions could save you a lot of time chasing down "*imagined*" problems.

There are two reasons just "binding everything" is avoided. First, there is the cost of performing a bind. A bind can use excessive resources, lock resources, and occasionally give you an access path you may not be totally thrilled with. If it was only 1 or 2 programs, no one would care. But a DB2 shop could have 100's, 1000's, even 10,000's of applications that need to be bound. The cost starts to add up. The second problem is the access paths themselves. No one wants to rebind a sub-second response time transaction and have it run in minutes or hours. Your results, the access path DB2 gives you when the bind completes, can be unpredictable. You are dealing with an enhanced optimizer that may be smarter, but may also be less forgiving. There is one potential solution though.

<sales_pitch_on>

IBM sells a tool, DB2 Path Checker for z/OS that may ease the pain and some of the fears of binding all of those applications. Path Checker can scan your DBRMs and tell you which one would get a different access path if they were bound. That means you would only have to bind the packages that would have a change in access path rather than binding everything. For details about this tool refer to the IBM DB2 Tools webpage at URL:

www.ibm.com/software/data/db2imstools/db2tools/db2pathchkr.html

</sales_pitch_off>

To quote my friend Roger Miller "To REBIND or not to REBIND, WHEN is the question. Whether 'tis nobler in the mind to bear the slings and arrows of outrageous access paths or to take arms against a sea of forces, and by rebinding, end them."

It takes work and will require staging in many customer shops, but with some care and work, customers can get almost all of the benefits of better access paths, with very few of the problems.

Plans, Packages and DBDs in DB2 Version 8

Rebind after upgrading to DB2 V8

Have you noticed a decrease in performance of some queries after migrating to DB2 for z/OS **Version 8**? I'm referring to those long running applications with SELECTs and FETCHs that process lots of rows and return lots of columns. Those queries ran just fine while you were in DB2 Version 7 and didn't seem to have problems until your V8 migration? Well, it just may not be your imagination!

First off, do you have all of your explain data for this application from your Version 7 environment? Did you also save this application's V7 accounting data so you have an idea how it might have actually run in V7 rather than depending on someone's recollection of how it performed? If you have checked out this application's explain data you captured after migrating to V8 and compared it to the saved data only to find out there is no difference in the access paths and the accounting data shows no new revelations, then a possible solution may be as simple as a REBIND.

How about I set the stage with a little bit of background first, then why a simple rebind will help may make more sense. You have all heard someone at sometime go through a dissertation about not using SELECT * or about minimizing the number of columns you specify on a SELECT. You get that speech because DB2 moves one column at a time between the DBM1 address space and the applications address space. The more columns you specify on a SELECT, especially columns you have no intention of ever referencing, the more it costs to execute that SELECT statement. However, sometimes you have no control over reducing the number columns specified or your application really needs all of those columns. Then what do you do?

Now we have to take a step back in time for a minute. Does anyone remember the release of DB2 Version 3? If you do, then you might also remember that DB2 V3 introduced a function called an SPROC. An SPROC, or SELECT procedure, is a mechanism that enables fast column processing, a performance enhancement that

examined a SELECT SQL statement that was executed repeatedly and built an internal procedure that moved all the columns in one move rather one column at a time via multiple moves. SPROCs just happen, you have no control over when or if. And the more columns that are specified on a SELECT, the greater the performance gain could be.

How does this all tie into Version 8 (finally)? If a plan or package is using an SPROC in V7, the SPROC is using 31 bit code. When you attempt to run that same plan or package in V8 without rebinding it first, it needs to be in 64 bit. It isn't, so DB2 disables the procedure. The only way you can re-enable the procedure is by rebinding it. Until you do that rebind, and if the plan or package uses an SPROC, your application's performance will be degraded. Do the rebind, and you should see a performance improvement.

The rebind can occur in compatibility mode (CM) or in new function mode (NFM). If you do rebind the plan or package in CM, there is no reason to bind it again in NFM because of the SPROC. When you move to V8 NFM and adjust some of the tables and indexes for better performance, then you'll need to rebind to get the next improvements.

As I have mentioned a few times already, you really need to get over your fear of binding once you get to V8. Almost all of the optimization enhancements are available to you in CM so why not take advantage of them? You should definitely rebind your plans and packages that have complex SQL. In addition, consider rebinding your packages and plans that have SQL that may be able to take advantage of index-only access from an index being changed to non-padded.

I am often asked what other optimizer enhancements would be available in new function mode (NFM) if a plan or package was already bound while in compatibility (CM). I guess I made the mistake in that post of saying that *most* optimization improvements were made available in CM. So I should have expected the "what happens in NFM" questions.

I used the word *most* intentionally. There is not a lot of documentation about what optimization enhancements are available in CM and which are available in NFM. However, what I can do is list some of the things that are documented as being available only after you get to NFM. I will take a little license as to what I think could, in some way, shape, or form, affect the optimizer.

Materialized query table (MQT) is a NFM only feature that lets you create an aggregate table that the optimizer can choose to use if certain conditions exist.

Multi-row processing for FETCH and INSERT discussed in two of my earlier blog entries
—

(blogs.ittoolbox.com/database/db2zos/archives/007377.asp

and

blogs.ittoolbox.com/database/db2zos/archives/007383.asp)

are SQL enhancements, not really optimization improvements. However, once you move to NFM, you will want to code your application's SQL to take advantage of these enhancements to improve your FETCH and INSERT performance.

Then there is the new locking protocol level 2 for data sharing. You not only have to be in NFM, you have to recycle (quiesce) your entire data sharing group to take advantage of it. And that makes perfect sense when you think about it. You probably don't want one member of your data sharing group using one locking protocol while another member uses a different protocol.

Mismatched numeric types and lengths were stage 2 in V7 and prior releases of DB2. Both of these conditions became stage 1 and indexable in Version 8. There is a bit of a story behind this enhancement though. When V8 was first delivered, you had to be in new function mode (NFM) to take advantage of this optimization change. A few months ago, thanks to APAR PK12389, this particular enhancement has been made available in compatibility mode (CM).

If you would like DB2 ODBC to support SQL statements up to 2 MB when connected to a DB2 for z/OS Version 8 subsystem, you will have to be in NFM and have applied APAR PQ88582.

If you have any DB2 ODBC applications that use an array INSERT in DB2 V7, the CLI/ODBC driver converts the array INSERT into individual row INSERTs. The new multi-row syntax will be used automatically for this once you move to new function mode (NFM).

Once you are in new function mode (NFM) you will be able to use multiple different encoding schemes in the same SQL statement. The biggest improvements for optimization are provided with the new clustering and index options. Index only access depends upon the NOT PADDED option for indexes. The new IS NOT DISTINCT predicate is stage 1 and indexable, while the prior construct included OR logic, which made it stage 2.

I am sure the above is not all inclusive. There are probably other enhancements that could affect how the optimizer makes its decision. For now though, you have a sample.

Bottom line: you really want to build into your migration plan a rebind strategy for all of your packages (and plans if you have plans with DBRMs bound into them directly) once you complete your move to NFM. Rebind will ensure that all packages (and plans) are in the correct format for your new catalog.

DSNTEP4

Another new function mode (NFM) feature you can look forward to taking advantage of almost immediately is the new sample program DSNTEP4. DSNTEP4 is an upgraded flavor of the well established sample program DSNTEP2. Written in PL/I just as its predecessor, DSNTEP4 gives you multi-row fetch capabilities. By simply setting the parameter MULT_FETCH to a value greater than 1 gives any SELECT statement the ability to fetch more than one row at a time. The use of this feature can greatly reduce the CPU time used by DSNTEP4 and could possibly reduce its elapse time. Remember though, you do have to be in V8 NFM to use DSNTEP4 because of the multi-row fetch

feature. You also want to make sure you stay current on maintenance for both DSNTEP4 and multi-row fetch. There have been a few APARs opened against DSNTEP4 and multi-row fetch in their short life that need to be applied before they can be successfully used.

DSNTEP2 (and its follow-on DSNTEP4) can also process the new V8 DIAGNOSIS SQL statement, the longer SQL table and column names, and SQL statements greater than 32K. You might have to be careful about warning messages with DSNTEP4 and the V8 flavor of DSNTEP2. They both do a much better job of reporting warnings in Version 8.

Because DSNTEP4 may be cheaper to run, in CPU terms, than DSNTEP2, you may want to consider making it your “standard” for running batch SQL. To make that kind of transition even easier, you may also want to consider doing the bind using DSNTEP4 to DSNTEP2 to avoid having to make all kind of JCL changes. If you do, make sure you remember that you made such a change so that any maintenance and future changes by IBM to DSNTEP2 or DSNTEP4 are made to the correct sample program.

If you need a description of the DSNTEP4 parameters or control cards, check out Appendix D of the “DB2 Utility Guide and Reference”, SC18-7427.

DSNTIAUL

The sample unload program is used extensively in some accounts. When you move to NFM, you should get the updated sample, replacing the old one. Where there are many rows to FETCH, the new program, using multirow FETCH, can reduce the CPU time by about half.

Caution: SPT (SCT) Directory Table Space Size Increase

What about the side effects of doing a package bind in Version 8? The side effect could be an increase in size for the SPT01 package table in the directory. It cannot exceed 64 GB. Have you checked the size of SPT01 recently? You should. And make note of how big it is. If it is greater than, let's say, 40Gb, you may not have enough room to rebind all of your packages in V8. Rebind (and bind) of a V7 package in V8 can result in a significantly larger package. You could see increases in size of packages by as much as 50%. How do you get around this potential issue? You don't. What you need to do is start cleaning up your packages. Are you using package versioning? If so, you want to start there.

How do you make a DB2 DBD dizzy?

The short, cute answer would be to migrate to Version 8. Just think: first you're V7 and then you're V8 and then you're V7 and then you're V8 and then you're... well, I think you get the idea. Now, before you start to think that I'm the one that is a little dizzy, how about if I explain where I'm heading and why I started out on this journey.

A friend of mine, Joan Keemle with John Deere, was researching a question for one of her friends. She went to Jay Yothers (IBM SVL, DB2 for z/OS Development) for some clarification. If you have a Version 8 question, Jay is one of the best places to start. Joan got the answer she was looking for and then thought it would be a good idea to share Jay's answer. She forwarded his reply to me and here we are... Isn't it nice when a plan comes together! (BY THE WAY, who said that?) So, I'm not really writing this entry, I'm just passing along some information.

The original question had to do with expanding DBDs in Version 8. Here is Jay's reply:

** V8 requires that the CCSID information in the DBD be accurate. CCSID information in DBDs written prior to V8 is sometimes OK, sometimes not ... but in the end, we can't rely on it, so when we load a DBD (in any mode) that was written prior to V8, we ensure the CCSID information is correct and write it back out. When we correct the CCSID info, we mark the DBD so that we don't do it again, since it isn't free. If such a DBD is subsequently altered and written out by V7, due to fallback or coexistence, that mark will disappear and we'll correct the CCSID info and write the DBD out again the next time it is loaded by V8.*

** When we read a DBD written prior to V8, we transform it to conform to V8 format.*

** When we write a DBD in CM, we transform it into V7 format.*

When you take those things together, you'll see that we could end up with a DBD in V7 format with the CCSIDs corrected that would stay that way, without some form of alter to it in ENFM or NFM. Doing a -DISPLAY DATABASE() in ENFM or NFM will cause all the DBDs not yet loaded in V8 to have their CCSID info corrected and written out in V8 format. But, these DBDs are probably less interesting than those used in CM, which would already have been written out in V7 format. Since this info is in the directory, there is no query you can do to find out which is which.*

The cost of transforming a DBD from V7 format to V8 format is in the realm of performance measurement noise. This means you can't notice it in the real world, but it isn't free. So, I would suggest making some innocuous alter to the DBDs you depend on in NFM at your leisure. You know which ones they are because you know their names without having to look them up. Just alter the primary space of one of the table spaces or indexes. Better yet, alter them to use our sliding secondary so that you don't have to worry about primary or secondary any more.

There you have it, a great answer to an often misunderstood concept.

Sometime before starting your upgrade to Version 8, or even while you are in compatibility mode (CM) but before doing any binds, check out the size of your SPT01 and SCT02 objects. I'm guessing most of you are primarily package shops. At least I hope you are. Plans without a PKLIST should be a rarity in this day and age. But I digress.

Take a look at the size of SPT01. SPT01 is a directory table space used to store skeleton package tables (SKPTs). If it is in the 30-40 GB range (or greater), you may be setting yourself up for some interesting CM fun. Remember, you are now in 64 bit mode. So when you bind a package, you could be using larger instructions, addresses, registers and control blocks, resulting in packages that could sometimes be as much as 50% larger. If you do the math, it doesn't take long to realize that if every package doubles in size and your SPT01 is already more than 32 GB, you might end up exceeding the 64 GB maximum size for a table space. If we are talking about packages and you do have a fairly large SPT01, see if you have a lot of excess package versions. If you are not using package versions, try to determine if you have packages that are no longer in use that you can free up.

Whatever you do, you want to avoid hitting the SPT01 max file page set size.

While you're thinking about packages, there is another item you might want to consider. You already know that the V7 EDM pool has been broken into 3 different pools in Version 8. A pool for DBDs (EDMDBDC) and a pool for the dynamic statement cache (EDMSTMTC) were moved above the bar. However, the EDMPOOL, used now for just plans and packages (CT, PT, SKCT, and SKPT) remained below the 2 GB bar. Your first impulse will probably be to decrease the size of the EDM pool because there is less in it. I think you should leave it alone, for at least a little while. Plans and packages will be larger in V8 after they are rebound. I would wait until I completed all of my rebinds and had a chance to determine the actual EDM pool utilization BEFORE reducing its size.

One final, kind of minor, thought. The DESCRIBE WITH STATIC option on installation panel DSNTIP4 (DSNZPARM parameter DESCSTAT on macro DSN6SPRM) has had its default change to YES in V8. This can lead to an increase in a package's size, increasing the storage requirements of SCT01 and possibly the EDM pool below the bar.

More on BIND: This time in regards to V8 NFM

I am a bit of a control freak. I like to know when things are going to happen and have some control (or least have the impression of control) over when they happen. Along those same lines, I really don't like surprises. It drives me nuts when something happens that I had no idea was going to happen. So the couple of "I got you" items when moving up to new function mode (NFM) are things I would prefer to avoid. Hopefully, a few of you think the same way.

The first is the auto rebind of invalidated packages. Yup, it drives me nuts when stuff just happens, especially binds. Because we all know that if a package is going to go through rebind, it is going to occur at the most inappropriate time. What am I talking about? The fact that if you drop an object that a package is dependent on, something like a table for example, the package will go invalid. I know you don't plan on dropping any objects during your move to NFM, but DB2 does.

During enabling new function mode (ENFM), DB2 will convert most of the catalog tables to Unicode. I said most. SYSCOPY for example, does not get converted. Another table that doesn't get converted is SYSDUMMY1. SYSIBM.SYSCOPY resides in its own table space DSNDB01.ISYSCOPY so it isn't an issue. However, SYSIBM.SYSDUMMY1 does not. It lives in DSNDB06.SYSSTR with a bunch of other tables that do get converted to Unicode. So, during ENFM, SYSDUMMY1 gets moved to DSNDB06.SYSEBCDC, a new table space added by DB2 in Version 8. Now go back to the previous paragraph. How do you move a table to a different table space? You drop it and recreate it, that's how.

You would be well advised, if you dislike surprise auto rebinds, to search SYSPACKDEP and find any packages that have a dependency on SYSDUMMY1. I would build a rebind job that includes all of the packages you find that reference it, and run that job at the completion of the REORGs that convert the catalog to Unicode. That would put you back in control.

Then there is that second little thing ENFM does to you. It turns off DATA CAPTURE CHANGES (resets it to NONE) for all catalog tables that you may have had it set on for. The best part, the ENFM job does NOT turn it back on. That little task is your responsibility at the end of ENFM or just after you move up to NFM. A quick catalog query against the DATACAPTURE column in SYSIBM.SYSTABLES will list all of the objects that will need altering at the completion of ENFM.

Last, but not least, there is the STATSTIME column in the SYSCOLUMNS catalog table. ENFM will invalidate values in this column.

Enabling New Function Mode (ENFM) Overview

So far the discussion has been about pre-migration planning and compatibility mode (CM). Now it's time to examine some of the things that happen in enabling new function mode (ENFM).

Enabling new function mode is the catalog conversion to Unicode step. ENFM is also the step in the Version 8 migration process that does not allow for any real fallback to Version 7. Yes, there is a way to get back to V7, but it is involved and far more complicated than you want to hear about in this paper. For the sake of this discussion, I would like you to just assume that when you make the jump to ENFM, it is "damn the torpedoes and full speed ahead" as they say. At this point I believe it is best to take a "move forward only strategy". Once you do start the ENFM step, complete the process, do some testing, and move on. With the exception of the catalog changes, you have tested most of this code already back in CM and most of the functionality that you are moving to V8 to take advantage of is not available to you until you complete your migration to new function mode (NFM). So make your visit to ENFM a short one. Do what you have to do, and move on the NFM.

Making the move to ENFM could be considered the most "interesting (??)" step in the migration to Version 8, as you will see in a minute. It is this step that I believe has the potential for causing the greatest migration anxiety. ENFM, among other things, converts the DB2 catalog to Unicode. And when someone hears Unicode, for some reason it just seems to raise their blood pressure. Unicode will be addressed shortly in its own section. For now, I just want to focus on what happens in ENFM.

Before you can even consider moving to ENFM, all of the members of your data sharing group (assuming you are running in a data sharing environment) must be running in compatibility mode; there can be no more coexistence. For data sharing or non data sharing environments, the catalog and directory need to be at a point of consistency before converting them to Unicode. The DB2 Quiesce utility can be used to establish that point of consistency. You should also avoid any kind of updates to the catalog or directory while in ENFM.

If everything is in V8 CM and a point of consistency has been established, an image copy of all of the catalog and directory table spaces should be run. When the copies are complete, it's time to run the installation job DSNTIJNE. This is the job that will position your catalog for running in full function DB2 Version 8. Besides the Unicode conversion step for each catalog table space, this job stream will also contain a step for each of the following functions on each of the catalog table spaces: save the current RBA or LRSN into the boot strap dataset (BSDS), make all of the necessary types and length changes

to the existing catalog columns (more on this in a minute), convert the catalog data to Unicode (next paragraph), change the buffer pools used by certain catalog table spaces because of page size increases (more coming up), and change catalog indexes built on varying length columns to NOT PADDED (additional details later).

I just stated that the DB2 catalog, or at least most of the catalog, is converted to Unicode. 17 of the 22 catalog table spaces and 1 directory table space will be converted. I need to emphasize that **ONLY** objects in the DB2 catalog and directory are converted to Unicode. This step absolutely does **NOT** touch any user data or user defined objects. DB2 is only going to convert stuff that DB2 owns. There are 17 catalog table spaces (in database DSND06) and 1 directory table space (in database DSND01) that will be converted to Unicode. They are: SYSDBASE, SYSDBAUT, SYSDDF, SYSGPAUT, SYSGROUP, SYSGRTNS, SYSHIST, SYSJAVA, SYSOBJ, SYSPKAGE, SYSPLAN, SYSSEQ, SYSSEQ2, SYSSTATS, SYSSTR, SYSUSER, SYSVIEWS, and SPT01 (directory). Of course, it might be easier to list the catalog table spaces that are not converted. SYSCOPY, SYSEBCDC, SYSJAUXA, and SYSJAUXB will remain in EBCDIC at the end of ENFM. Of the directory table spaces, the 4 that will remain in EBCDIC are: DBD01, SCT02, SYSLGRNX, and SYSUTILX. Also keep in mind that, once the catalog has been converted to Unicode, the collating sequence is different than EBCDIC, so some of your catalog query results can change.

As mentioned previously in this paper, the DB2 REORG utility performs the Unicode conversion. You want to make sure that you have run REORG against the catalog long before you get to ENFM. Running the REORG during compatibility mode, the first time that REORG has been available to run against the **entire** DB2 catalog, gives you practice running REORG, gives you some sample timings so you will have some idea how long ENFM will take, and gives you a nice orderly catalog that should improve the performance of actually doing the Unicode conversion.

Suggestion: *You may want to consider making a copy of the DB2 catalog and using it for practice. You will not only be able to test how REORG converts your catalog, you will also be able to develop some timings to help predict how long the process will take on your actual catalog.*

A few of the columns in the catalog will have length increases and data type changes because of the implementation of long names in V8. Long names in V8 are going to be defined in the catalog as VARCHAR (128). You can see immediately how this is going to affect everything you do DB2. Just something as simple as a column that participates in an index key could be challenging. Now you can see the importance of the new NOT PADDED being forced to the catalog indexes by the ENFM step.

So what columns in the catalog have won this increase in length? In most cases, that is a pretty straightforward answer. However, there are some interesting exceptions that I will discuss. But first, what type of columns will become VARCHAR (128)? Here is the list as of the last time I played with a DB2 V8 catalog. This list may not be all inclusive but it should give you a flavor for the breadth of this V8 enhancement. DB2 defines the following column types as VARCHAR (128): tables, views, aliases, synonyms, indexes, schemas, columns*, constraints, storage groups, collections, packages*, authorization ids*, locations*, distinct types, routines (user defined functions, cast functions, or stored procedures), triggers, JAR files, sequence names, and identities. Quite a list isn't it? Every application you have written and every vendor tool you have purchased that

references the DB2 catalog will probably have to be changed. Almost every SQL statement you have tucked away to query the catalog will more than likely also have to be changed. Imagine doing a SELECT in SPUFI against SYSIBM.SYSTABLES with a few of its columns now defined as VARCHAR (128) and what the returned output might look like.

Did you notice how I flagged a few items with an asterisk (*)? These are the exceptions I was referring to earlier. Columns, although defined in the catalog as VARCHAR (128), only allow a maximum length of 30. There are still things that have to be changed in DB2 before a column will reach its full potential. I am guessing someone at SVL found some problems for applications with the longer column length. The SQLDA column length is 30 bytes. So, they gave you as much increase as they could without giving you application shock. Next in my list is package name. Everyplace a package name is used in the catalog has been changed to a VARCHAR (128). Packages created as a result of issuing the CREATE TRIGGER statement can take advantage of all 128 bytes. However, any package created via the BIND command will still have a maximum length of 8 characters. Authorization ids still have ties to other things in z/OS and they will remain with a max length of 8 bytes, although the catalog says VARCHAR (128). And finally, locations. Locations will still have a 16 byte limit, even though the catalog defines all columns associated with a location as VARCHAR (128). Those are the exceptions that I know of. If there are others, please let me know.

I mentioned earlier that the length of some columns change in ENFM causing a page size change to something greater than 4K. With a larger page size comes the need to use a different buffer pool. With that, a few of the catalog table spaces are modified job DSNTIJNE to use buffer pools other than BP0. The catalog table spaces SYSDBASE, SYSGRTNS, SYSHIST, SYSOBJ, SYSSTR, and SYSVIEWS will all have an 8K page size in V8 and use buffer pool BP8K0. The page size for catalog table space SYSSTATS will increase to 16K and it will use BP16K0 in Version 8.

The last thing I mentioned was the NOT PADDED feature. NOT PADDED is added to all of the catalog indexes during ENFM to avoid any problem with longer keys being built on VARCHAR (128) columns. This is an "I gotcha" in this process though. If you are using user defined indexes on the catalog, shadow datasets must be added to the DSNTIJNE job stream. This job has no problem converting user defined indexes to Unicode if they have been added to the job stream. The shadow datasets need to have space for PADDED indexes. However, if you do not take this action prior to running the job, DSNTIJNE will fail. After DSNTIJNE, alter the indexes to NOT PADDED. An alternative solution would be to drop all of your user defined indexes, run DSNTIJNE, and then recreate the user defined index structures specifying NOT PADDED after DSNTIJNE has completed successfully.

If you are curious about where you are in the ENFM process, you can use the DB2 command DISPLAY GROUP DETAIL to find out. The DSN7101I portion of the DISPLAY output has a MODE keyword. MODE can have 5 possible values: C – compatibility mode, E – enabling new function mode, and N – new function mode; it can also be set to C* which means you are in compatibility mode but have been at a higher level and E* meaning you are at enabling new function mode but have been to a higher mode. You will see more about E* and C* when DB2 9 is delivered. This display command will also give you the status of which table spaces have been converted to Unicode in the DSN7133I message.

By the way, while you are in ENFM, your database request modules (DBRM) are still going to be created using EBCDIC. You do not end up with Unicode DBRMs until you complete your migration to new function mode (NFM) and set the precompiler option to NEWFUN(YES).

It is important to understand, and well worth repeating here, that regardless of which "mode" of your V8 migration you are in, they are all still the same DB2 code base. There is only one code base for DB2 for z/OS Version 8. CM is the same code base as ENFM and ENFM is the same code base as NFM. The question of code base changes came up at a recent user group meeting. I was surprised that there was confusion about this before finding out that this question comes up quite a bit. I believe that the way we explain the V8 migration process, it sounds like you might be running a different set of code in CM (compatibility mode) from what you run in ENFM or NFM. This is absolutely not true. All of the 64 bit stuff and DB2's use of Unicode is there in CM. In fact, see the earlier section on bind where I discuss the Unicode SQL parser.

Let's revisit ENFM for a minute

I was listening to someone describe their V8 migration and they pointed out a few potential trouble spots that so far I have failed to mention. So what else might happen during enabling new function mode (ENFM) of the V8 upgrade. ENFM is the phase that converts the catalog table spaces to Unicode and long names and the catalog indexes to "not padded" along with a whole bunch of other stuff. But you already know all about that from earlier in this document. Now, I would like to point out a few things that "might" get you into trouble, if you are not careful.

One of the more significant reasons (IMHO) for the ENFM job stream (DSNTIJNE) that does the Unicode conversion (and other stuff) failing is lack of adequate sort space and/or the allocation of the correct size shadow tables. You may want to consider checking the job before submitting to ensure that the allocation values all seem reasonable. Pay specific attention to SPT01 and SCT02. They can tend to be problematic when it comes to allocating the correct sizes. You may also want to change SORTNUM from 4 to 64 for the steps that deal with SPT01 and SCT02.

Everyone likes to be especially cautious during a migration like V8, so you should also consider running a DSN1COPY CHECK and DSN1CHKR before running any of the ENFM jobs to revalidate the catalog. This idea may take on even more importance if you have been hanging around in compatibility mode (CM) for the last 6 or so months. You are converting most of the catalog to Unicode and long names using DB2's REORG utility. You really do not want to find a bad link or corrupted page during the middle of your catalog migration.

Don't forget to run RUNSTATS on the DB2 catalog when you finish running the catalog migration jobs. Statistics are invalidated during the migration. Statistics for EBCDIC values are not the same as those for Unicode UTF-8.

And while you are checking things out before running your ENFM jobs, you should make another check to ensure that DATACAPTURE is set to NO for the catalog tables and that you have done something about all of the user defined indexes you have created on the catalog. If the ENFM job that does the Unicode conversion runs into a user defined

index without a shadow dataset or inadequate space while processing the catalog for migration, it will stop the migration job dead in its tracks.

DB2 Version 8 and Unicode

Unicode is an interesting subject. For some reason there still seems to be a lot of confusion on the subject. Unicode is one of those things that looks new yet has been around for a long time. Although it has been an optional part of DB2 on the OS/390 and z/OS platforms since Version 6, it didn't get our attention (the DB2 for z/OS user community) until when we found out that parts of DB2 would start to use Unicode and the "optional" part was going away. As you know, during ENFM, the DB2 catalog is converted from EBCDIC to Unicode. There are other areas of DB2 that Unicode plays a significant role in that we will discuss in just a few minutes. These are the reasons that just bringing up the subject of Unicode during a discussion can get a group's heart rate racing. These are also the reasons why at every conference I have attended since V8's use of Unicode was announced, it can be difficult to get into one of Chris Crone's sessions on Unicode if you hadn't planned to arrive at least 20 minutes early (Chris, by the way, is IBM SVL's resident expert on Unicode). There is also way too much stuff around Unicode to discuss in this paper. So, today we will simply try to describe what Unicode is, why Unicode might cause you some concerns, and point you to a lot of resources so you can read more about Unicode should this entry spark your interest.

What is Unicode?

Before going into too much more detail about Unicode, I should stop and explain a few terms that you will start to see in Unicode discussions. You are probably used to dealing with EBCDIC and are quite familiar with the fact that a hexadecimal "F1" represents a numeric "1", a hex "C1" represents the upper case character "A", and the hex value "81" represents a lower case character "a". These hex representations of your data are called code points. When you group a bunch of these code points together, they are referred to as code pages. If you have been around z/OS for a while (even going back to the early releases of MVS), I'm sure you have run into code page issues. Code pages have been around since the dinosaur. By the way, code pages are **NOT** the same as CCSIDs. Do not interchange the two terms, they really are different. A CCSID (Coded Character Set Identifier) is just a numeric value that *identifies* a code page. CCSIDs are part of the metadata that describes the data in a DB2 subsystem and I believe they were first introduced to DB2 back in Version 4. For example, if you are using a US English code page, it will be represented by CCSID 037. There is also a special case US English code page that includes the euro symbol that has a CCSID 1140. Both CCSIDs 037 and 1140 will probably be valid for most US shops. A CCSID in DB2 is also specified for the entire DB2 subsystem and once specified, really should not be modified. That "should not be modified" phrase is what could cause you concern when you start your DB2 V8 planning & migration.

Of course, the big question is "What is Unicode?" The official definition from the Unicode web site (www.unicode.org) is: *"Unicode is the universal character encoding, maintained by the Unicode Consortium (www.unicode.org/consortium/consort.html). This encoding standard provides the basis for processing, storage and interchange of text data in any language in all modern software and information technology protocols.* A quote from the Unicode website that I like even better is: *"Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language".* I think this quote really sums up why Unicode is important.

Everyone gets to work with the same character representation, end of story. With Unicode standard 4.0, there are currently over 64,000 characters identified by unique numeric codes.

When only code pages were in use, different languages would use different codes pages. In fact, there were often occasions where a single language might use multiple code pages. The result can be character translation errors. Unicode came into existence to minimize these problems. Unicode attempts to represent every possible character by its own Unicode code point. To accomplish this, Unicode uses Universal Transformation Formats (UTFs). UTF not only includes code points for most languages, it also covers math and science symbols. This format also allows for easy inclusion of new characters.

Should Unicode be of concern?

To find out if you should be concerned, your first stop should be DSNHDECP to validate what CCSID you currently have DB2 set up to use. For customers that should be using a US English CCSID, the error most frequently discovered is CCSID 500 or, in some rare cases, CCSID 0.

The DSNHDECP module contains your application programming defaults, among them the CCSID that is in use. CCSID 500 is usually associated with Belgium, Switzerland, and international Latin. CCSID 037 is usually associated with countries using US English. In some instances CCSID 1140 may be specified in place of CCSID 037 if the euro symbol is being used. The other CCSID values specified in DSNHDECP are listed in the following table.

DSNHDECP keyword	Your Current CCSID value	Description
SCCSID		single-byte EBCDIC
MCCSID		mixed EBCDIC
GCCSID		graphic EBCDIC
ASCCSID		single-byte ASCII
AMCCSID		mixed ASCII
AGCCSID		graphic ASCII
USCCSID		single-byte Unicode
UMCCSID		mixed Unicode
UGCCSID		graphic Unicode

Could you have a Unicode concern? Maybe, maybe not. A determination cannot be made by simply running a query against the DB2 catalog. There are two tasks that should now be performed. First, run job DSNTIJP8. This job is delivered via APAR PQ84421 (PTF UQ85439 on RSU 0406). It is the DB2 Version 8 flavor of DSNTIJP8. Do not run DSNTIJP8 delivered with DB2 Version 7. This is the verification job for migrating from Version 6 to Version 7. The Version 8 flavor of DSNTIJP8 is delivered on the Version 8 distribution tapes. This job should be run on your DB2 Version 7 at your earliest convenience to identify any concerns that need to be resolved before commencing a migration to V8. Any CCSID issues should be completely resolved **BEFORE** you start your migration to DB2 V8 compatibility mode.

If you discover you may not be using the CCSID you should be using, should you change it? **Absolutely not!** After running DSNTIJP8, if you have CCSID concerns,

you should open an ETR with DB2 Level 2 Support specifying DB2 **“CCSID 500 prior to migrating to Version 8”** as the subject line. Level 2 will provide you with the instructions on how to proceed. You should not change any occurrences of CCSID 500 or CCSID 0 (if those are the CCSIDs you are concerned about) in the DB2 catalog or in DSNHDECP without guidance from DB2 Level 2 Support. I strongly recommend that an ETR be opened as soon as possible if you discover you are not using the CCSID you think you should be using. This action should be taken if your Version 8 migration is not planned for some time. This will give you adequate time to make changes, should it be determined that changes are necessary.

The possible problem (emphasizing possible), that may arise when having a CCSID specified that does not match the data is the possibility of improper translation of some characters. Here is an example. DB2 thinks a piece of data is encoded using CCSID 500 although DB2 Version 7 is not really enforcing that CCSID and the data appears to really be using CCSID 037. When the Version 8 migration is complete, the DB2 catalog will be encoded using Unicode. If you were using “[“ (left square bracket) it would have had the EBCDIC hex representation of “BA”. DB2 would encode that character using the appropriate Unicode value. When that character is subsequently retrieved from DB2, and your DB2 is now Version 8 NFM or ENFM, the character will be converted from Unicode back to CCSID 500 because DB2 is now enforcing the CCISD. Now the character returned will be a right corner, a character not on my keyboard. A character translation error has just occurred. Using CCISD 037 and 500 as examples and because CCSID 500 seems to be the CCSID mis-coded in most cases, there are actually 7 characters that have different hex values as demonstrated in the following table.

Code Points	Character Comparisons			
	CCSID 037		CCSID 500	
4A	cent sign	¢	left bracket	[
4F	logical or		exclamation point	!
5A	Exclamation point	!	right bracket]
5F	logical not	¬	circumflex accent	^
B0	circumflex accent	^	cent sign	¢
BA	left bracket	[logical not	¬
BB	right bracket]	logical or	

It should be pointed out that there is data that is unaffected by Unicode. For example, unaffected would be any column defined as “FOR BIT DATA”, and numeric data stored as binary, packed, or float.

One concern often mentioned in a first time Unicode discussion is the amount of space the catalog will take after encoding it in Unicode. Customers often express a concern over a significant increase in catalog size. When the catalog is converted to Unicode, how much the catalog might increase in size is dependent on the type of data being converted in the catalog. For example, if your current CCSID is 037 then each character will take up a single byte. Almost all of the characters in CCSID 037, with few exceptions, will still only take up a single byte when encoded in Unicode. So your increase would be minimal, if not zero. If you are using a lot of special characters and/or double byte characters, then the Unicode equivalents will probably use more space than the EBCDIC character.

Another area that may cause some concern is the collating sequence used once you complete the conversion to Unicode. The first 127 Unicode characters are the same as ASCII so the collating sequence will be different. The collating sequence for Unicode is numeric, uppercase characters, and then lower case characters (1, 2, 3, A, B, C, a, b, c). In EBCDIC, the collating sequence is lower case, upper case, and then numeric (a, b, c, A, B, C, 1, 2, 3). This will affect both data ordering and the results from range predicates. Once your conversion of the catalog to Unicode is complete, you may need to validate any in-house queries that are used against the catalog, and any tools that process the information in the catalog, to check you still get the same results.

What else could be affected? I have heard the comment that “Unicode only changes the catalog so the affect on me should be minimal”. Not true. Beside have the catalog in Unicode and affecting collating sequence and range predicates, you need to be aware that predicate evaluation and SQL parsing are performed using Unicode. Once you complete your upgrade of Version 8 to new function mode (NFM) and NEWFUN(YES), all of your DBRMs will be created using Unicode, all SQL stored in SYSIBM.SYSSTMT and SYSIBM.SYSPACKSTMT is stored in Unicode, literals may be in Unicode, the results from an SQL statement may be Unicode, and SQL included in IFCIDs “could” be in Unicode. This last item has a ZPARM that controls the format.

Unicode Resources

What about those references? Here is a starter list of resources to get you going:

- There is a very nice two part article on Unicode from IBM's Sarah Ellis that you can download from IBM's support site. Just search on Sarah's name.
- As mentioned earlier, there are two must read informational APARs containing all of the latest and greatest information you will need to use Unicode with DB2 Version 8. They are II13048: Support for Unicode: Steps for Installing and Customizing (Part 1) and II13049: Support for Unicode: Steps for Installing and Customizing (Part 2).
- For some general information concerning your DB2 Version 7 to Version 8 migration and fallback, you should reference APAR II13695.
- I have also discussed on a few occasions an APAR available for DB2 Version 7 that will allow you to check out your DB2 catalog and help you identify any possible (or potential) Unicode problems. PQ84421: Add DSNTIJP8 - V8 Pre Migration Checkout Job - TO V7.
- There is an excellent presentation (with notes) by Chris Crone that you should definitely take a look at if you have Unicode questions. Check out Unicode and DB2 V8, What You Need to Know on the DB2 support site.
- Since I am mentioning white pages and presentations, go to the IBM DB2 support web page, www.ibm.com/software/data/db2/zos/support.html and do a search on Unicode. You will get pages of Redbooks, presentations, white papers, articles, and a list of current APARs for and about Unicode.
- Finally, Unicode has its own web site if you are interested in a “pure” description not written by someone at IBM. Their site is located at www.unicode.org/.

Unicode and IFCIDs

Let's discuss IFCIDs (Instrumentation Facility Component Identifier) for a minute. This is definitely a subject, in my humble opinion that needs a little more explanation. Once you have completed your migration to DB2 for z/OS Version 8, your catalog will have

been converted to Unicode. With that, any SQL information added to certain IFCIDs could also be formatted in Unicode. "Could" is the operative word for this entry.

DB2 Version 8 introduces a new DSNZPARM keyword that controls how certain information is coded when it is included in an IFCID. The new ZPARM, UIFCIDS, is a keyword on the DSN6SYSP macro. It is initially set at installation time using the "UNICODE IFCIDS" entry (line 11 in V8) on panel DSNTIPB. Your possible choices are NO, the default, and of course YES. If you specify NO, information added to the IFCIDs remains in EBCDIC format as in previous versions of DB2. If YES is specified, then some of the information added to IFCIDs will be in Unicode. NO seems like a more logical choice, allowing the information to remain in EBCDIC format, and therefore making it "easier" to deal with. However, if you specify NO, DB2 must convert that information, which is now being processed by DB2 using Unicode, into EBCDIC. Although that cost may seem slight, it is still a cost. With the number of IFCIDs that some customers cut, well, it's just something to think about.

What I would strongly suggest is that you consider specifying **UIFCIDS=YES**. With YES, the information is simply added to the appropriate IFCID, no conversion required, in Unicode format. The information that remains in Unicode format is only a subset of the total information written out to an IFCID, and the field that is written out to an IFCID using Unicode is marked with a %U. By specifying YES, the "cost" of converting from Unicode to EBCDIC is moved to the monitoring tool rather than the DB2 subsystem. Any decent tool should be able to handle Unicode in a DB2 Version 8 environment.

CCSIDs, Unicode, and DB2 V7, Oh no!

The title sounds a little like Dorothy's line from the Wizard of Oz: "Lions and tigers and bears, Oh my!" Although CCSIDs, Unicode, and DB2 V7 in the same sentence are something you definitely need to be concerned about, I'm not sure they are quite as scary as Dorothy's plight on the Yellow Brick Road.

Here's the scope and the potential issue. If you are still on DB2 Version 7 you should be planning for your V8 upgrade. During your planning, and most importantly while you are still in DB2 V7, one very critical item that really needs to be taken care of is the health and feeding of your DB2's CCSIDs. It is imperative that you ensure your CCSID values in your DB2 V7 subsystem are correct (meaning the CCSIDs are what you expect them to be). Your CCSID needs to be valid, it can't be 0 (zero), and your DB2 subsystem should only have one. Right now you might be thinking "How does one know if all of the above is true?" That part's easy.

RSU 0406 delivered the medicine that could relieve your CCSID concerns with PTF UQ85439 (APAR PQ84421). This PTF contains job DSNTIJP8, a jobstream made up of a set of catalog queries that will identify "situations" in the catalog that need to be addressed before upgrading to Version 8. Some of the queries in this job have something to do with validating the CCSIDs in your Version 7 DB2 subsystem

You want to run this job on V7 and run it soon. If you do discover that your CCSIDs are not what you expect, I would strongly suggest that you open up an ETR (we call them PMRs) with DB2 Level 2 Support. Specify in the heading that this problem has to do with CCSIDs and your V8 upgrade. There is a team that specializes in repairing these kinds of issues. You do NOT want to just start changing stuff before talking to Level 2.

And you **ABSOLUTELY** want to take care of any CCSID issue while you are still in DB2 Version 7. Once you get to V8 NFM, your catalog will be in Unicode. You want to have any and all CCSID problems resolved long before you reach that point. **If you have a CCSID issue, it will be far easier to remedy it while in DB2 Version 7 then it will ever be to fix it in Version 8.**

These queries are actually also delivered with DB2 for z/OS Version 8 in job DSNTIJPM. However, to have access to this job you would have to install V8, which kind of defeats the whole purpose of using this query to check things out prior to upgrading to V8.

When upgrading to Version 8 or planning your upgrade, one of the most important things to remember is to read everything you can find from IBM about a V8 upgrade and follow every instruction and suggestion. I can assure you that doing a little extra planning on the front end could save you from having a lot of problems on the back end. With that in mind, when you are pulling maintenance and doing all of this pre-planning stuff while in Version7, make sure you pull APAR II13695. This informational APAR contains a lot of good information necessary for successfully pre-planning your V8 upgrade. While you are at it, you should also pull the informational APARs for Unicode. They also make for a great read prior to upgrading. The Unicode APARs are II13048 (Part I) and II13049 (Part II).

Enabling New Function Mode (ENFM)

DB2 for z/OS Version 8 Long Names

So, you've just finished running DSNTIJNE moving your DB2 subsystem or data sharing group into enabling new function mode (ENFM). You also read some dude's blog that said all you need to do is run a few queries to make sure the catalog was still there and survived the conversion from EBCDIC to Unicode. So, you jump into SPUFI and run a SELECT * FROM SYSIBM.SYSPACKAGE and you get a package name and lots of blanks... "Oh no, what's this?".... "Oh, wait a minute, things are probably OK.." You forgot about the conversation to long names. Well, you didn't forget about, you just really never gave what it may mean a second thought.

One of the multiple tasks performed by DSNTIJNE while in enabling new function mode (ENFM) is the conversion of a bunch of "names" in the catalog to what we have been referring to as long names. Columns, mostly variable or fixed length and 18 characters long are converted to VARCHAR columns with a maximum length of 128 characters. But what are 100 characters among friends, right? As we tried to demonstrate in the first paragraph, it could be a lot.

There are actually two completely different changes mentioned above, each bringing their own little concerns and both lumped under "long names". Yes, the length has increased from 18 to 128 characters for many columns that name stuff. "Was 18 in V7" is the operative phrase here. Anything displaying one of these columns needs to be aware of the longer lengths, even doing something as simple as running a query in SPUFI. If you are not accustomed to using SUBSTR or SUBSTRING (using length units specified like CODEUNITS32, CODEUNITS16, or OCTETS) functions, then it is definitely time to learn. You could be seeing a lot of these if you right your own catalog queries. You may also want to check out the LENGTH function, just in case you ever have a need to know how long these fields are before you use it. And then there is that

VARCHAR thing. Not only do you have to code for length, now you have to actually extract the length so your application gets it all correct.

Speaking of SPUFI and longer lengths on some columns, there may be a few adjustments you might want to consider making. On the current SPUFI defaults panel, you may want to consider changing MAX CHAR FIELD to something longer than 128, otherwise a lot of columns that now use long names may get truncated from the column's right hand side. Also keep in mind that you can control the width of the output from SPUFI that is displayed by ISPF BROWSE. By default, the record length and block size in SPUFI is set to 4096. If you reduce these values by making the display narrower, columns could wrap. Remember, you always have the option to scroll left and right in ISPF BROWSE.

What columns are affected by long names in Version 8? Quite a few! The biggies (in my opinion) are alias, auxiliary table, collection id, constraint name, correlation name, distinct type function, index, JARs, procedure, schema, sequence, specific, statement, storage group, synonym, table, trigger, and view. There are more, but I just listed a few so I could demonstrate the potential impact here.

A couple of names that did not change are the columns for table space and database. They both still have to participate in the underlying VSAM objects and z/OS has a whole different set of restrictions. Still, the names are in Unicode now, and Unicode for some characters is larger, so the column definitions are changed to VARCHAR(24). There are often trailing blanks in these VARCHAR columns, so you may need to adjust SQL that uses LIKE predicates. And there are a couple of "got'chas" columns. For one, the name of a column in SYSIBM.SYSCOLUMNS. A column name in the catalog has a length defined as VARCHAR (128); however, DB2 still imposes a 30 character maximum restriction on column names. Another is package name. This column is also VARCHAR (128), but only a package created by CREATE TRIGGER can use the whole length. A package created via BIND is still only 8 characters. Authid's are also still 8 characters, although all of the CREATOR type columns are define as VARCHAR (128). Schema names may be in the CREATOR columns, and they can exceed 8 characters.

I am guessing that upon initially upgrading to V8, none of this will have much of an impact, other than the SUBSTR thing in SPUFI. Not many of you (can you say none) will have a lot of names longer the 18 characters before upgrading. But give it time. You will....

Another comment you may mumble under your breath the first time you dabble with the catalog after completing ENFM that has nothing to do with long names is the order of things returned. It's all going to seem pretty much backwards for a while. After all, your catalog is now encoded using Unicode, not EBCDIC. And with Unicode comes a whole new order to things. Everything is going to sort a lot like ASCII. After all, the first 127 Unicode characters are the same as the first 127 ASCII characters. But fear not, there is a possible solution for that also.

Let's say you wanted to list out all of your tables but you want them returned in the same order they would have been returned back in V7. To return the result of this query in EBCDIC collating sequence, use

```
SELECT CAST (NAME AS VARCHAR(128) CCSID EBCDIC) AS E_NAME
```

```
FROM SYSIBM.SYSTABLE  
WHERE.... <font color="#FF0000" size="+1"><b>*</b></font>
```

My last long name comment has to do with the PLAN_TABLE. There are a bunch of columns there that are also going to change to VARCHAR (128). Fortunately, you always get a sample PLAN_TABLE so this should not be a big deal, except of course for your queries that you use against the PLAN_TABLE.

New Function Mode (NFM) Overview

It's finally time to pull the big trigger and start using all of that DB2 Version 8 stuff you've heard so much about. The planning has been completed, you ran the IVPs and performed all of the testing under Compat Mode, and migrated the catalog during ENFM. Now you are ready to make that final move to new function mode (NFM). You have listened to all of the recommendations to only stay in ENFM for a short time. What can you expect and what should you now do?

Anytime after completing the run of DSNTIJNE, you are ready for your last step. DSNTIJNF does that final piece to move you to new function mode (NFM). This job runs the CATENFM utility. CATENFM is invoked by both DSNTIJNE and DSNTIJNF. The first time it is used is in job DSNTIJNE, specifying the START keyword to move DB2 into enabling new function mode. It is also called by DSNTIJNF with the COMPLETE keyword so DB2 can enter new function mode. When specifying COMPLETE, CATENFM will check to make sure DB2 is ready to move to NFM. It checks to make sure long name support has been added, all of the catalog indexes are created, available, and defined with NOT PADDED, and that all 18 table spaces have been converted to Unicode. Run DSNTIJNG to rebuild DSNHDECP changing the new keyword NEWFUN=YES defaulting the precompiler to process new functions. In fact, at this point, all DB2 for z/OS Version 8 features and functions are available. However, with the ability to use all that new V8 function, comes the elimination of migrating back to compatibility mode (or DB2 Version 7 for that matter), or the use of data sharing release coexistence (actually back in ENFM).

What's left to do? You can now start creating user defined NOT PADDED indexes on the DB2 catalog, delete the VSAM dataset for DSNDB06.DSNKCX01 (the index for the now finally departed SYSIBM.SYSPROCEDURES), and run DSNJCNVB (the conversion/expansion utility) to convert the BSDS (boot strap dataset) to handle 93 active logs and 10,000 archive logs. Running DSNJCNVB does require DB2 to be stopped. Once you convert to BSDS, you are ready to run the Version 8 sample jobs that are shipped with V8. Remember you ran the Version 7 samples while you were in compatibility mode. During ENFM, DATA CAPTURE is set to NONE for all catalog tables except SYSIBM.SYSCOPY. Once you have completed entering new function mode (NFM), you can use the SQL ALTER statement to re-enable DATA CAPTURE on the catalog. However, remember that you must be in new function mode before enabling DATA CAPTURE.

At some point after entering new function mode, you will also want to quiesce your data sharing group (assuming you are running data sharing of course) to enable Locking Protocol II. This will cause a data sharing outage so plan for it well in advance.

What happens if you decide that you are not quite as ready as you thought you were to have applications use all of the new Version 8 SQL? Can you return to ENFM? Sure. Just run job DSNTIJEN (CATENFM with the ENFMON keyword) to reenter ENFM and have DSNTIJNG rebuild DSNHDECP to specify NEWFUN=NO once again. NEWFUN is also a precompiler keyword and can be reset to NO at precompile time.

It may not seem like a big deal now, but you will be required to be in new function mode before you can migrate to DB2 9 for z/OS. I know there will be someone out there tempted to just stay in ENFM forever for some reason.

NFM is running, now what?

After all the work you went through getting to new function mode (NFM), what's the reward for your effort? Read on. What follows is nothing more than a laundry list of function delivered with NFM, and not a complete list at that.

Do you dabble with SQL? Of course you do. And V8 has a ton of new stuff for you, some of it was even delivered before DB2 for Linux™, UNIX® and Windows™ (luw). It's kind of cool getting something before the "kids" do. One of the most talked about improvements in this area is limit changes. Almost all of the DB2 objects (table, view, alias, synonym, trigger, index, etc) have had the lengths of their names increased from 18 to 128 characters, column lengths have been increased from 18 to 30 characters, index key size is now 2000, cursor names have gone from 18 to 30 characters, and 225 tables now be included in a join.

There is also a ton of new SQL in Version 8. The biggies, in no particular order, are dynamic scrollable cursors, multi-row FETCH and INSERT, common table expressions, recursive SQL, sequence objects, multiple distinct clauses, SELECT within an INSERT, scalar fullselect (a full SELECT within parentheses that can be used as an expression), expressions on the GROUP BY clause, session variables, new functions and special registers, and EXPLAIN from the statement cache.

The above is far from being all encompassing. There are lots of other SQL and SQL related enhancements along with availability, usability, performance, data sharing, and distributed, far more than I can do justice to in a this paper. However, almost all of them can be found in the IBM Redbook *"DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More"*, SG24-6079.

Here are a few more improvements that I believe you will find interesting:

- **NOT PADDED Indexes** - DB2 will store the data length with the key for indexes defined on variable length columns. Not only does this give better storage utilization for long keys, it allows index-only access on indexes with keys defined on variable length columns
- **Backwards Index Scanning** - yup, you can read an index backwards so there is no longer a need for two separate indexes with same with one defined as ascending and the other descending. One ascending index can handle the reads in both directions.
- **Unlike Types are now Indexable** - mismatched data types can now be stage 1 and indexable. This is huge for those application that do not support all of the data types available in DB2 for z/OS

- **Non-Uniform Distribution Statistics on Non-Indexed Columns** - in V7, you had DSTATS to column non-indexed column statistics to enhance the optimizers choice of access paths. That was good but it was an application running against your data. In V8, that functionality has been moved into RUNSTATS.
- **Partitioning and clustering separated** - you can now partition on one value while clustering on another
- **DB2 Family Compatible SQL** - Version 8 has greatly closed the gap between DB2 for z/OS and DB2 for luw. V8 actually beat DB2 luw to the punch for a few of the SQL statements being delivered.
- **Parallel Sorting** - V8 attempts to take advantage of parallel sorting more often.
- **64-bit Exploitation** - V8 takes advantage of 64-bit memory in many ways; pools above the bar, more storage for parallel task, dictionaries, etc...
- **Data Partitioned Secondary Indexes** - full partition independence for secondary indexes. Careful though, using DPSI may require SQL changes to maintain the wanted access path. By the way, non partitioned secondary indexes are now referred to as NPSI rather than simply NPI. We need to keep with the cute acronym theme.
- **4096 Partitions** - one partition for every day of an 11 year period. Now that is some serious size and growth for a single table.
- **2000 byte Keys** - index keys have been increased to 2000 byte max making it easier to define large amounts of data in the index of index-only access.
- **Long Object Names** - most object names in V8 have been increased to 128 bytes.
- **2 MB SQL Statements** - the 32 KB max SQL size in V7 was restrictive and prevented coding complex SQL and SQL stored procedures. This limitation has been removed in V8.
- **Code Page Independent Parsing** - DB2's SQL parser now uses Unicode. This eliminates the problems with code page differences in literals and other SQL elements.
- **Multiple Encodings** - ASCII, EBCDIC, and Unicode data can all be used in the same SQL statement.
- **Common Table Expressions** - a common table expression is like a temporary view that is defined within and for the use of a single SQL statement. This feature has been around DB2 for luw for a long time and now is finally available in DB2 for z/OS.
- **Recursive SQL** - remember the old bill-of-materials type application queries that we could never do on DB2 for z/OS? Well, now you can because of the previous enhancement (common table expressions).
- **Multiple DISTINCT** - multiple distinct can be specified for a single SQL statement.
- **Scalar Full Select** - a full SELECT statement that returns a single value can be used anywhere in an SQL expression where a column could be used in the past.
- **225 Table Joins** - 225 tables can now be specified in the FROM clause of a single SQL statement. Just think of the fun you can have debugging that query!
- **Materialized Query Tables** - AKA automatic summary tables or materialized views, allows the optimizer to rewrite a query to make use of pre-aggregated results set.
- **Multi-row FETCH and INSERT** - allows more than one row to be fetched or inserted by a single SQL statement. Distributed SQL will automatically take advantage of the FETCH portion of this enhancement.
- **Automatic Space Allocation** - AKA sliding secondary extents processing, allows DB2 to manage secondary, and in some cases primary, space allocations for table spaces.
- **More log space** - the active logs have been increased to 93 and the archive logs increased to 10,000.

- **Online schema change** - more table attributes can now be changed dynamically, without the need to drop and recreate the table.
- **System level point in time recovery** - you now have a SYSTEM BACKUP utility (requires z/OS V1.5 or above) and a SYSTEM RESTORE for faster recoverability of your entire subsystem or data sharing group.
- and more...

Why move to V8 NFM

Over the next few weeks, I will be making a series of short post with reasons that I believe should convince you to make that move to V8 new function mode (NFM). I know, I'm into this series thing. But the good news is it gives me a way to write some short post for those times when you don't have a lot of time to do a lot of reading.

While having discussions about DB2 for z/OS Version 8 with customers, I am often asked "Why move to V8 NFM?" Many customers want reasons to help then make that decision to complete the final big move. So this series of post will be intended to help... hopefully. I hope you find them helpful.

Once you finish your V8 upgrade to new function mode (NFM), something you will want to take advantage of almost immediately is the new sample program DSNTEP4. DSNTEP4 is an upgraded flavor of the well established sample program DSNTEP2. Written in PL/I just as its predecessor, DSNTEP4 gives you multi-row fetch capabilities. By simply setting the parameter MULT_FETCH to a value greater than 1 gives any SELECT statement the ability to fetch more than one row at a time. The use of this feature can greatly reduce the CPU time used by DSNTEP4 and could possibly reduce its elapse time. Remember though, you do have to be in V8 NFM to use DSNTEP4 because of the multi-row fetch feature. You also what to make sure you stay current on maintenance for both DSNTEP4 and multi-row fetch. There have been a few APARs opened against DSNTEP4 and multi-row fetch in their short life that need to applied before they can be successfully used.

DSNTEP2 (and its follow-on DSNTEP4) can also process the new V8 DIAGNOSIS SQL statement, the longer SQL table and column names, and SQL statements greater than 32K. You might have to be careful about warning messages with DSNTEP4 and the V8 flavor of DSNTEP2. They both do a much better job of reporting warnings in Version 8.

Because DSNTEP4 may be cheaper to run, in CPU terms, than DSNTEP2, you may want to consider making it your "standard" for running batch SQL. To make that kind of transition even easier, you may also want to consider doing the bind using DSNTEP4 to DSNTEP2 to avoid having to make all kind of JCL changes. If you do, make sure you remember that you made such a change so that any maintenance and future changes by IBM to DSNTEP2 or DSNTEP4 are made to the correct sample program.

If you need a description of the DSNTEP4 parameters or control cards, check out Appendix D of the "DB2 Utility Guide and Reference", (SC18-7427). Remember to also check the web for the latest versions of all DB2 documentation. When the manuals are changed, you will see those changes on the above web page first.

The improvements to DSNTEP2 and the introduction of DSNTEP4 are just a few more reasons why you want to move to NFM. They are also very strong examples why

staying in capability mode (CM) for extended periods of time may not be in your best interest. Although CM is the place you want to be to complete all of your V8 testing and, without a doubt, where you want to stay until you are comfortable and confident that moving forward is in your best interest, you do want to have it in your place to eventually move to NFM.

DSNTIAUL changes are similar to those in DSNTDP4. The function is the same, but when you fetch large numbers of rows, the CPU is reduced by about half. If you run DSNTIAUL a lot, then you can save a lot of the endangered CPU cycles.

The secrets to a successful upgrade to DB2 for z/OS Version 8

We try not letting many people in on the inside scope of how to successfully upgrade to DB2 Version 8. If everyone knew the secret, well you know what would happen. However, for this paper I'm going to break that rule and tell you that there are secrets around upgrading to DB2 V8 and minimizing the potential for problem. In fact, there are two huge secrets. (This is the point where you look around and make sure no one is reading over your shoulder; maybe even take this paper to a more private spot.... OK, is it safe....)

The really big secrets are:

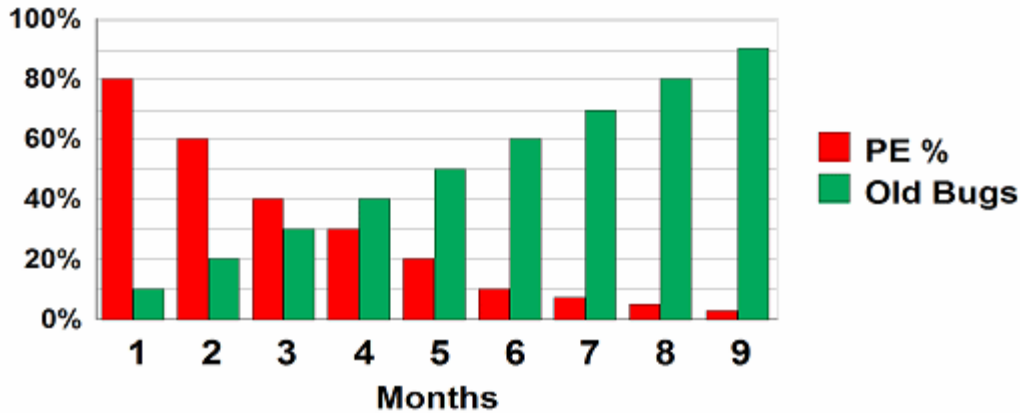
- **Planning**
- **Maintenance**

There, I said it. I let the cat out of the bag. Now everyone is going to have a better chance at success and there will be less reason for anyone to read my blog. Gee, this could get depressing..... (smile)

I have been pushing the planning stuff since I started my blog back in July 2005. There are probably a couple dozen posts there that discuss different aspects of planning and its importance. (So, maybe planning is not as big a secret as I thought). But even with all those posts, I still get questions about making the upgrade process go smoother. Do you know that there are still some out there that never even think of reading the manual? It's simply blast in and start to install, install, install until it's installed correctly.

The second big secret may be a bit more laudable yet a whole lot more controversial. Although most believe planning prior to starting an upgrade to V8 will lead to a higher chance of success (even though many will not actually put into practice a simple step that they know will definitely help), telling someone to stay current on their maintenance just flat out bugs a lot of people. It sometimes seems as though so many have such a long list of "good reasons" why they just can't possibly apply maintenance to their DB2 on any kind of regular basis. It's kind of sad that so many feel that way. That maintenance thing could be as easy to accomplish and as rewarding, as the planning thing.

Check out this graph (more information courtesy of Roger Miller)...



The decreasing red bars represent the potential number of PTFs⁴ that could be in error, in month order, by the month that the PTF was delivered. The increasing green bars are the number of reported DB2 problems that already have a PTF available to fix the problem. Notice what happens around month 5 & 6. The number of problems that could have been avoided if maintenance had only been applied far exceeds the number of potential problems that might have occurred if that same maintenance was applied. Every time you hit a problem in the 5 or greater months, it's an outage that could have been avoided if only DB2 maintenance had been applied in a more timely fashion. We have preached for years (almost for the entire life cycle of DB2) that you are far better off putting maintenance on than you are avoiding it. The above chart attempts to graphically dramatize that. My strong recommendation is to try to stay about 5-6 months current on PTF plus applying as many hipers on a weekly basis (or what ever frequent schedule you can afford) to keep your DB2 subsystem up and running. I know how frustrating it can be to have an outage that requires you to open a PMR and ship tons of documentation to IBM to figure the problem out. Well, let me tell you that it can be just as frustrating to us to spend days analyzing dumps and traces only to find out the entire issue could have been avoided if a few PTFs had been applied. It is really aggravating when the fix turns out to be something with a hiper available.

As with all secret things there must also be magic words, or at least that's the way it works in my version of the world. I have two magic words (actually two magic acronyms) that fit this discussion perfectly. They are: CST (Consolidated Service Test) and RSU (Recommended Service Upgrade)⁵. When chanted together, and with a certain amount of magic dust thrown in for effect, they will form something that we in DB2 land like to call "**preventative service**". Yup, by just applying a few PTFs, you "*could*" save yourself a ton of potential grief and possibly someday having to deal with a needless PMR (ETR). Although DB2 Level 2 Support is a wonderful group of people, they are probably one of the few groups in IBM that would be completely thrilled if they never had the occasion to talk to you.

⁴ PTF - Program Temporary Fix – This acronym is worth giving special mention to because I love the term. APAR (or authorized program analysis report) is the actual temporary piece. The PTF (temporary in its name) is actually the permanent part of a fix. OK, maybe it isn't all that funny now, but on a Friday at midnight it's a complete panic to think about.

⁵ For details about CST and RSU maintenance, refer to the IBM website:
www.ibm.com/servers/eserver/zseries/zos/servicetst/mission.html

Why mention CST and RSU? RSU is an already aged and tested set of PTFs ready to install. You can apply an RSU tape with a high degree of confidence because they have been tested for your z/OS environment. Because an RSU tape contains a quarters tested PTFs, by waiting one additional quarter, all the PTFs (as mentioned before already tested in a similar z/OS environment) fall into the 5+ month range mentioned earlier.

I have just one last comment about SMP/E before I close out this section. With SMP/E V3.4 you can take advantage of something called SMP/E Internet Service Retrieval. This feature allows you to acquire your maintenance electronically at any interval you want, so that the service you need is always there waiting for you.

What's your DB2 for z/OS V8 upgrade strategy?

An upgrade strategy (*the part that tells you how you're going to get from V7 to V8 new function mode (NFM)*), in my opinion, is almost as personal as how one fixes their bagel in the morning. As long as you understand the "how comes" for choosing the upgrade strategy you chose, most should be successful. What I am going to attempt in the following paragraphs is to describe how I might proceed on the path to Version 8. Now, you all did notice the disclaimer warning you that this is my opinion; nothing more, nothing less. For now, I am just attempting to explain what decisions I would make in the hope that you will go through a similar process when you start to build your strategy or plan.

Step one: You need a plan. I don't care if it is a good plan or something bazaar. Show me something I can follow, something I can comment on, something that lets me know you have given some thought on how to get from point A to point B. It can be anything from a simple Word document or Excel spreadsheet to a full blown work plan using a project management tool. You just have to something written down.

Step two: You need to write your plan down and use that written plan as your working document. That plan will tell you where you have been, where you are, and where you should be heading. That plan gives you something to discuss and to modify. Without that plan, you are simply winging it on a prayer and a song, hoping all will turn out in some positive manner once all of the dust settles.

Step three: Well, there isn't a step three. This is where I actually start to describe my plan.

We all know about the three modes you have go through in order to make it to a full blown really cool DB2 for z/OS Version 8 subsystem. And this is where the confusion begins.

You have compatibility mode (CM), enabling new function mode (ENFM), and new function mode (NFM). It seems pretty straight forward at first, but if you start to over analyze each of the modes during the planning process I can almost guarantee your head might explode. This brings me to the reason for making this post in the first place. To describe how I would progress through the three magical, mystical, mysterious phases on my way to V8 euphoria?

We all know what the three phases are: compatibility mode (CM), enabling new function mode (ENFM), and new function mode (NFM). You have all been told this is the same stuff you have always done in the past, only now it's enforced and has names. You will test in CM, move quickly through ENFM, and end up (eventually, and hopefully sooner than later) in NFM. It all sounds so simple when someone says it. Just keep in mind while you read this post what I used to tell my customers when I did consulting. I give great advice, I have spectacular ideas, and I know just how you should do everything, I have all of the answers. Oh ya, my pager never goes off at 3:00AM because something breaks. In other words, this post is just an example of a starting point. So, let's take a look at four steps to migrate to V8.... Did I say four? Just wait...

First stop: compatibility mode (CM). You have installed DB2 for z/OS Version 8 and you are now ready to start testing to see that the installation was successful. That's what we are doing in CM. You are here to test the DB2 Version 8 code. You are not here to test some new piece of SQL that will eventually be available in Version 8. You are in CM to make sure DB2 works correctly. Remember, the DBM1 and IRLM are now running in 64 bit mode. While you are in CM, all of the Version 8 code is there. The different modes do not run different flavors of DB2 code. Sure, different features are made available during different modes, but it's all still the same piece of code. So, beat on it for a while, put it through its paces, and do what ever you can think off in an effort to break it. That's why you are in CM, to make sure the V8 base code works correctly. This is the mode, in my opinion, that you want to spend the most time. Heck, this is where you want to spend all of the time. You want to exercise V8 in every possible way you can think of. If you are one of those customers that like to run IVPs, run it. Just remember you have to use the Version 7 IVP stuff while in CM. The Version 8 IVP will not work.

You want to make sure you run CM through at least one major event. You know, end of month, end of quarter, or end of year if that is the most important event you go through. It doesn't really matter what the event is, as long as you believe it is the event the runs the majority of your most significant applications. You are also getting most of the optimization improvements while you are in CM mode. Which brings up the question: why are you afraid to bind? One of the new features of V8 that you must test is the new optimizer stuff. You don't want to wait until ENFM or NFM to find out you have an application that will not perform because once you make it to ENFM or NFM, falling back to CM or Version 7 are no longer options. You are, in what I call, a fall forward strategy. Fix it until it works. But we are getting ahead of ourselves, aren't we. We are still in CM, right. And in CM you want to run all of that old SQL to see that it runs the same in V8 as it did in V7. You want to compare EXPLAIN data from V8 and V7, you want to check out accounting reports from V8 and V7. This is the only way of telling if you are having a performance issue. Telling your end users you are now in V8 and asking them how things are running is not an accurate or an adequate measurement. The mere fact they know you changed something will taint their judgment which may cause them to give you a negative, although not intentionally negative, estimation of the situation.

While you are in V8 CM you need to run everything you can find that ran in V7, you want to test everything you can locate to test. You want to make sure you have no V8 issues of any kind (to the best of your knowledge at least) while you are still in compatibility mode (CM). You can only really fall back to Version 7 from CM. And yes, I know that is not a 100% true statement but it is a sentiment that few would disagree with.

Suggestion: *If you attempt a rebind in V8 and it is not successful, please check and make sure you could rebind that same package (or plan) in Version 7 before yelling for help. There is nothing worse than trying to shoot a problem that really does not exist.*

Now it is time for step 1.5: the fourth, and almost invisible, step. This step is for data sharing shops only and is referred to as coexistence. Coexistence allows you to run one data sharing member in Version 7 while another data sharing member is in Version 8. It is term used to describe a way of moving parts of your data sharing environment forward while leaving other parts behind. All I can really say here is that coexistence should be completed in days or weeks, not months. It is not a stage that anyone intended anyone to stay in for an extended period of time. Migrate a few members one weekend, migrate the rest the next weekend, and get it over with and move on.

Moving forward to step 3: enabling new function mode (ENFM). Think about what this step really does. It converts the DB2 catalog to Unicode, long names and padded indexes. That is basically all that is going to happen. You have already successfully tested all of the V8 code, excluding new features, so there is really no need to spend anytime in ENFM. Convert the catalog, validate the conversion was successful, and move forward. What else is there for you to do. Planning to do extensive testing in ENFM makes no sense because there is nothing left to test except for the catalog migration. You want to move through ENFM quickly and get into NFM. When you make the decision to go from CM to ENFM, you have already decided that V8 works to you expectations. You tested it in CM. Other than testing the catalog migration, there is nothing in ENFM that is new. It is just CM with a new catalog structure.

The final step; new function mode (NFM). You have tested all you can test in CM. All that is left to do now is make the new features in V8 available to all to take advantage of. When you move to NFM, that is what you are doing. You are turning on the new stuff. You are setting some bits in the directory and changing NEWFUN to YES in DECP. You have tested all you can test and now it's time to find out how all of that new stuff works. NFM is the ultimate fall forward mode. If you were to have a problem while in NFM, you open a PMR and get it fixed. If however, you decide for some reason you really don't want anyone messing with any of the new stuff, you can return to ENFM and disable all of those new NFM features.

These are just a few thoughts on how I would do things, given the change of course. But I am not doing your migration and I do not have to live with the consequences. With that in mind, take all of this with a grain of salt or make it the catalyst for a team discussion of your potential migration plan. But whatever you do, just make sure you have a plan or strategy, or whatever I have been calling it in this post.

One closing thought. Give a lot of consideration to what order you move your systems. You may want to migrate them backwards; meaning move your production subsystem to NFM first, and then migrate your other systems. Why? You want to make sure that what ever subsystem is feeding another subsystem (for example: test to QA, QA to stress test, stress test to production), can't pass an incompatible feature forward.

Countdown to the end of DB2's Private Protocol

The people at DB2 Development have been talking about DB2's private protocol going away for what seems like forever. Can you say "No changes since DB2 Version 4"?

That's a heck of a long time to go with enhancements. And all through V5, V6, V7, and now V8 the subtle suggestions have kept on coming that you need to start migrating your applications that use private protocol to Distributed Relational Database Architecture™ (DRDA). Up until now, your only incentive has been the lack of change for 10 years and the promise that someday private protocol will be removed.

Well, it looks like the promise is about to be fulfilled. DB2 9 will bring you DB2's next step towards private protocol's demise. If you attempt to use private protocol when binding or rebinding a plan or package (*DBPROTOCOL(PRIVATE)* keyword) once you have upgraded to DB2 9, a warning message will be generated. This is to help you identify what plans and packages are not using DRDA.

DB2 9 does take some steps to ease your transition from private protocol to DRDA. Until DB2 9, you had to use SNA to get support for "Already Verified" through the network. TCP/IP did not support it. "Already Verified" has been one of the reasons some have used for remaining with private protocol. Fortunately in DB2 9, Roles and Trusted Context may give you a similar function. You will be able to define remote SNA connection as a trusted IP address and emulate "Already Verified".

For a long time, three-part name support and DRDA's performance were also often considered inhibitors to moving off of private protocol. However, back in DB2 V6 three-part name support was added to DRDA so an application could take advantage of DRDA without having to be re-written. As for those rumors of performance issues? They were also resolved long ago. DRDA can use static SQL, while private protocol is dynamic. So DRDA generally performs much better than private protocol. Other than a few brand-x applications, there really is little reason to still be on private protocol.

If you have any plans or packages still using private protocol, start planning today to get rid of them. Regardless of the version of DB2 you are currently running on, finding which plans and packages not playing the DRDA game is pretty easy; maybe the easiest part of this entire process. You just need to query the DBPROTOCOL columns in SYSIBM.SYSPACKAGE and SYSIBM.SYSPLAN for all occurrences where DBPROTOCOL is equal to **P (private protocol)** or not equal to **D (DRDA)**.

DB2 9 will give you even more assistance. An APAR, PK27413, with the associated PTF UK16431 (that became available July 26, 2006) makes available a REXX exec, DSNTP2D9, that will query the DB2 catalog and create the REBIND control cards to rebind the package locally and the BIND control cards to bind the package at any necessary remote locations. DSNTP2D9 will also create the SQL CONNECT, CREATE ALIAS, and RELEASE statements to set up the correct aliases at the remote locations. Currently PTF UK16431 is only available for DB2 9.

An update on moving off of DB2's private protocol

I mentioned a REXX exec being shipped with a DB2 9 APAR that could assist a customer's final move off of private protocol. What is today's good news? That REXX EXEC delivered in that DB2 9 APAR has been made available to you all in flavors that will run with DB2 for OS/390 & z/OS Version 7 and DB2 for z/OS Version 8. You have no need to wait until you upgrade to DB2 9 to start getting off of private protocol.

You can download the appropriate EXEC for your particular environment from the IBM DB2 Support website and search on "***Determining if plans or packages have a remote location dependency***". Keep in mind that there are **TWO** different EXECs on this web page. There is one EXEC for V7 and one for V8. Make sure you download the correct one for your DB2.

Two REXX EXECs are supplied: DSNTP2D7 and DSNTP2D8. You are also supplied with the JCL to run each EXEC in batch. The JCL is included inline as comments in the EXEC along with installation and run instructions. As I just mentioned, these EXECs are written to run in very specific environments. DSNTP2D7 will **ONLY** work with DB2 Version 7 or DB2 Version 8 compatibility mode (CM). DSNTP2D8 is written to **ONLY** run in DB2 V8 new function mode (NFM). If you are running V8, make sure you use the correct EXEC. And obviously, you do need DB2 REXX Language support to use these EXECs.

From the documentation in the REXX EXEC: *This program has one required input parameter and four optional parameters. Based on the input parameters, this program examines the catalog of a DB2 subsystem to generate necessary commands to convert plans and packages that refer to objects in remote DB2 locations from using private protocol communications to access those objects to instead use DRDA protocol communications. One side effect of converting to DRDA protocol is that any SQL statement that makes a reference to a remote object via an ALIAS in the local DB2 will not have the ALIAS reference within the statement resolved before the statement is sent on to the remote server. Thus, another function of this program is to examine the defined ALIASes within the DB2 subsystem and generate any necessary two-part name CREATE ALIAS statements which then should be executed at the location referenced by the local ALIAS.*

The following is taken directly from the above web page.

Abstract

These sample REXX programs analyze the DB2 subsystem catalog to see if any plans or packages that were bound with DBPROTOCOL(PRIVATE) have a remote location dependency. If they do, the tool generates commands to convert those plans or packages to use DRDA protocol. Also, the programs analyze any three-part name aliases in the catalog to determine if corresponding two-part name aliases must be created at the indicated remote locations.

What's your incentive?

There are some excellent reasons to start your V8 upgrade today. Here are just a few:

- DB2 Version 7 is going out of service. We all knew it would happen eventually. However, now IBM has made it official announcing an end-of-marketing (EoM) and end-of-service (EoS) date. On the IBM Software Support Lifecycle web site, the EoM for DB2 V7 is March 5, 2007 (announcement letter 906-254) and the EoS will be June 30, 2008 (announcement letter 907-023). That means EoS is just a little more than a year away. With planning, your V7 to V8 migration could easily take 6-12 months. By the way, if you are still running a version of DB2 older than V7, then you are already running code well out of service. Just think of all the neat stuff you are missing out on.

Just for completeness, DB2 Version 7 went generally available (GA) on March 30, 2001, with announcement letter 201-054.

- IBM has delivered DB2 9. It has been discussed at conferences and in the press for a long time. We know that you will have to be in DB2 Version 8 new function mode (NFM) and z/OS 1.7 before even thinking of upgrading.
DB2 was announced March 6, 2007. Details of that announcement can be found the IBM United States Software Announcement 207-041.
- DB2 9 includes full pureXML™ support
- Only DB2 Version 8 (and above) is enabled to the recently announced System z9 Integrated Information Processor (zIIP) specialty engine. A portion of DB2 V8 work will be eligible to take advantage of this engine. If you are still on V7, even if you have a z9, you cannot.
- Speaking of a z9, examples of some of the z9 and z/OS features that DB2 can take advantage of are:
 - Hardware Compression
 - System z9 Integrated Information Processor (zIIP)
 - System z Application Assist Processors (zAAP)
 - Integrated Facility for Linux (IFL)
 - Internal Coupling Facility (ICF)
 - FICON channel and MIDAW
 - z/Architecture instruction set
 - Parallel Sysplex
 - z/OS Workload Manager (WLM)
 - Hipersockets
 - IPv6 (DB2 9)
 - Integrated Cryptographic Service Facility (ICSF)
 - Secure Sockets Layer (SSL) (DB2 9)
 - Kerberos V5, Public Key infrastructure
 - Multilevel security (RACF)
 - CP Assist for Cryptographic Function (CPACF)
 - Peripheral Component Interconnect Extended Cryptographic Coprocessor (PCIXCC)
 - Microcode-assisted sort
 - UNIX System Services
 - Dynamic virtual IP addressing (VIPA)
 - Intelligent Resource Director (IRD)
 - Parallel Access Volumes (PAV) and Multiple Allegiance
 - ESS FlashCopy
 - I/O Request Priority (IORP)
 - VSAM data striping
 - Geographically Dispersed Parallel Sysplex (GDPS)
 - Peer to Peer Remote Copy (PPRC)
 - Extended Remote Copy (XRC).
 - HyperSwap
 - And much, much more...

Migration/Fallback Info APAR for DB2 9

So, how's you upgrade to DB2 for z/OS Version 8 going? Do you have your plan in place, are you in compatibility mode (CM), or maybe you have moved to new function mode (NFM)? Heck, maybe you're just feeling adventurous today. If you are, then you really need check out this latest DB2 APAR: II12323: DB2 V8.1 MIGRATION/FALLBACK INFOAPAR TO/FROM DB2 V9.1.

You're reading that correctly. It's the Migration/Fallback APAR to get to DB2 9 for z/OS and it's now available, complete with a list of APARs with their associated PTFs that have already closed. Pretty cool, right!!!!

Oh ya, when you're checking this new APAR out, you may find a second DB2 9 Migration/Fallback APAR out there. Ignore it. We will get it down to just one APAR number shortly. Once you are running in DB2 V8 NFM, you should probably get the DB2 9 checking or premigration job DSNTIJP9 (APAR PK31841) and look in the DB2 9 Installation Guide for the list of incompatible changes. Some of the problems can be avoided or changed more gradually.

About the Author

William Favero is currently an IBM Senior Certified IT Software Specialist, the DB2 for z/OS Sales Specialist for the West Region in IBM's Sales and Distribution organization, and an IBM Certified Solutions Expert for DB2 for z/OS Version 8, for DB2 UDB, and for DB2 for OS/390 Version 7 and part of IBM's zChampions team. He has over 30 years experience working with database and more than 22 years of that working primarily with the DB2 family. Willie is a highly sought after speaker on a variety of DB2 for z/OS topics for numerous conferences and user groups and frequently publishes articles in various database magazines. You can contact Willie at:

wfavero@us.ibm.com

Copyright Notice

©Copyright IBM Corporation 2006

IBM Corporation
1133 Westchester Ave.
White Plains, NY 10604
Produced in the United States of America
02-06
All Rights Reserved

IBM reserves the right to change specifications or other product information without notice. This publication could include technical inaccuracies or typographical errors. References herein to IBM products and services do not imply that IBM intends to make them available in other countries. IBM provides this publication "as is" without warranty or condition of any kind, either express or implied, including the implied warranties or conditions of merchantability or fitness for a particular purpose. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions; therefore this disclaimer may not apply to you.

IBM makes no representations or warranties regarding third-party products or services. Some software may differ from its retail version (if available); and may not include user manuals or all program functionality. For non-IBM software, applicable third-party software licenses may apply.

Information about non-IBM products was obtained from suppliers of those products. Questions concerning those products should be directed to those suppliers.

IBM, the IBM logo, the e-business logo, DB2 Universal Database MVS/ESA, MVS, OS/390, WebSphere, z/OS and zSeries are trademarks of IBM Corporation in the United States, other countries, or both. Microsoft is a trademark of Microsoft Corporation.

Windows and Windows NT are registered trademarks of Microsoft Corporation.

Java is a trademark of Sun Microsystems, Inc.

UNIX is a registered trademark of the Open Company in the United States and other countries.

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