DB2 for z/OS Version 8: What we learned from upgrade planning.

August 2007

Contents

Introduction

It has been more than three years since IBM DB2 Universal Database™ (UDB) 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 that planning is essential to the success of an upgrade. 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 right 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.

When you first hear the upgrade process to Version 8 explained, it might sound like IBM has come up with a new and possibly confusing way to get from Version 7 to Version 8. However, the process that you follow during migration to Version 8 is not much different from the best-practice process you have been following since DB2® Version 1. The only difference is that the process has now been formalized. In the past, you updated the IBM 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.

You follow that same process when moving to Version 8. First you upgrade the DB2 catalog using CATMAINT and perform all initial testing of the new code, just like in the past. That step is called *compatibility mode* (CM), and new SQL function (plus a few other features) is not yet available. CM is also the only time you can fall back to Version 7. Compatibility mode is a conversion mode intended to make the transition as simple as possible. Incompatible changes are introduced at this point, so that you can fall back to Version 7 easily. However, new function that is compatible with the previous release is not yet introduced; this prevents you from using a feature or function during migration that might prevent you from falling back to Version 7. After you are comfortable that the catalog migration works and the new version behaves as advertised, the next

step is to convert the catalog to Unicode and enable the use of long names. This step is new and a little different compared to previous releases of DB2. It's called *enabling new-function mode* (ENFM). After you move to ENFM, you can no longer fall back to Version 7; you can only "fall" forward. Finally, you need 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 nearly all features in Version 8. You can move safely back and forth between ENFM and NFM modes, enabling and disabling the use of new features and functions.

That's the view from 60,000 feet of upgrading to DB2 for z/OS Version 8. The rest of this paper will go into more detail.

Reference material

When you are migrating to Version 8, you need to read everything you can find from IBM about Version 8 and follow every instruction and suggestion. Doing a little extra work up front can save you from having a lot of problems later. With respect to migrating to DB2 for z/OS Version 8, some reference material is as important as the discussion of migration, if not more so. This section lists where to find the information that you can use during migration.

The most important DB2 Web site to bookmark is the IBM DB2 for z/OS home page at **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 information becomes available, so plan on visiting often to stay up-to-date.

Another frequently visited DB2 Web page is the IBM DB2 Support page, at **ibm.com**/software/data/db2/zos/support.html. This page provides an easy and straightforward way to search for just about anything related to DB2, including white papers, articles, IBM Redbooks® publications, presentations given by IBM, answers to questions, and information about PTFs and APARs.

For a complete description of the RSU maintenance process, see **ibm.com**/servers/eserver/zseries/zos/servicetst/mission.html.

No discussion about DB2 materials is complete without a list of current Redbooks publications. Here are some that are invaluable to anyone working with Version 8:

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 Version 8
SG24-6480-01	Securing 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 Version 8: 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 publications*

A Web site that is critical to your success with DB2 for z/OS Version 8 is the page containing all of the DB2 Version 8 reference manuals and product documentation. Located at **ibm.com**/software/data/db2/zos/Version 8books.html, this page provides the latest versions of all of the Version 8 product manuals, program directories, and extender documentation. The only manuals not included on this Web page are the licensed materials. This is another one of those pages you can bookmark so you can always make sure you are referencing the latest DB2 information. Be careful whenever

using hardcopy versions of the product reference manuals, because you never know how old they are or what might have just been changed. Always check the Web for the latest updates. When you are planning for DB2 Version 8 or testing in Version 8 compatibility mode, be aware that the product documentation generally assumes that you are in new function mode (NFM). If a manual refers to a feature or function that pertains only in compatibility mode, that restriction is generally stated explicitly. However, in general, the DB2 Version 8 documentation is talking about NFM.

Before the product tapes arrive

Let's start with a discussion of preparing for migration before actually explaining the different migration modes in Version 8. You've just returned from the International DB2 User's Group, the DB2 Tech Conference, SHARE, or maybe a local database user group, and you are enthusiastic about moving to DB2 for z/OS Version 8. While you are waiting for the decision to be made about when to order the new DB2 and when to install it, use this time to do some analysis and planning to prepare for the migration to Version 8.

Hardware prerequisites

First, determine if your company is properly positioned to consider a Version 8 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 the IBM z/Architecture® instruction set. Your options include the IBM z800, z890, z900 (with the proper microcode), z990, z9 Business Class (BC) and z9™ Enterprise Class (EC) processors. You also need to ensure that you have adequate real storage to support z/OS, DB2 and any applications required to run in the same environment.

Operating system prerequisites

When DB2 Version 8 requires a minimum release level of z/OS V1.5 for certain functions delivered in Version 8, don't consider a release that is not current. In addition, you might want to think about other operating system considerations. For example, if you are thinking about taking advantage of the IBM System z™ (formerly zSeries®) Application Assist Processor (zAAP) or System z9™ Integrated Information Processor (zIIP) specialty engines, z/OS V1.6 (or later)

is the minimum operating system required to use them. The zAAP specialty engine is for Java™ virtual machine (JVM) workload and the zIIP specialty engine is used by DB2 Version 8. When considering z/OS V1.6, keep in mind that it is scheduled to go out of service in the Fall of 2007, and so it will be out of service soon. If you can choose the operating system on which you should install DB2 Version 8 and want to be as current as possible, consider z/OS V1.7 or V1.8.

Migration and fallback support

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® and z/OS Version 7. You cannot skip DB2 versions allowed this time. You also need to make sure that the maintenance installed for DB2 Version 7 is up-to-date. The required migration and fallback support is built into the Version 7 maintenance stream. The Version 7 PTF UQ81009, the DB2 Version 8 fallback-toleration PTF, is included as part of the Recommended Service Upgrade (RSU)[1] maintenance on RSU0403. You must apply this PTF to DB2 Version 7 before starting your Version 8 migration. After applying the PTF, you must restart the DB2 Version 7 subsystem before starting DB2 Version 8. If a DB2 data-sharing environment will be migrated to Version 8, this PTF must be applied to all members of a data-sharing group before DB2 Version 8 is started on any member of that data-sharing group. This APAR is critical. It allows a subsystem that has been migrated to DB2 Version 8 compatibility mode to fall back to DB2 Version 7 if problems occur with Version 8. Beginning with DB2 Version 8, the fallback-toleration maintenance is no longer optional. It must be applied or the migration process fails.

Suggestion: Before starting any software migration, consider scheduling at least one DB2 outage, regardless of whether you have data sharing. When the fallback-toleration PTF is applied, the DB2 subsystems must be restarted before Version 8 can be installed. In a data-sharing environment, this service can be applied without stopping all of the subsystems.

Validate the DB2 catalog

Next, check whether the DB2 catalog is ready for migration. Accomplishing this should be easy if you have stayed current on maintenance. That's because the fix for APAR PQ84421 (PTF UQ85439), delivered with RSU 0406, makes available job DSNTIJP8, which contains a set of catalog queries that can be used to identify situations in the catalog that need to be addressed before migrating to DB2 Version 8. You must run this job.

Monitor informational APARs throughout the upgrade

There are multiple informational APARs that you should monitor throughout the life of the Version 8 upgrade process. APAR II13695 summarizes all of the latest migration information in one place. This APAR includes required and suggested maintenance along with migration hints and tips. Check for updates to this APAR through all three phases of the Version 8 migration process. In addition, there are two Unicode informational APARs that you should also refer to before and during the migration: II13048 (Part I) and II13049 (Part II).

Take an inventory of DB2 tools

Another essential task can be time consuming and maybe not quite as straightforward as you might hope: taking an inventory of the tools you use to support DB2. As soon as possible, start making a list of all the products you have in 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. 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 work with or on DB2. After you have built that list of tools contact the vendors to find out what you have to do to prepare their products for migrating to Version 8. You need to verify that any tool you plan on using with DB2 Version 8 works with Version 8. Changes related to 64-bit support, long names, Unicode, and other enhancements might be potential problems for some DB2 tools. Be wary of products you

have had for many years, or products that are out of service but are still in use. Fortunately, if you are using any of the IBM tools that support DB2 for z/OS, they are already enabled for Version 8. For more information about these IBM tools, check the IBM DB2 and IMS™ Tools page at **ibm.com**/software/data/db2imstools/.

Prepare to run the IVP (for DB2 Version 7)

Another premigration task you should consider is preparing to run the installation verification procedure (IVP) for DB2 Version 7. Although a set of sample databases and IVP jobs are installed during the process of migrating to and implementing DB2 Version 8, they are designed to validate the DB2 Version 8 installation and migration processes after you complete the move to new function mode, not while you are still in compatibility mode. So how can you validate that your migration to Version 8 compatibility mode worked as planned? You need to an IVP. However, in this case, you need the sample databases and IVPs from DB2 Version 7. If you no longer have them on disk, you must rerun portions of the DB2 Version 7 installation CLIST to reinstall them.

Clean up and reorganize the DB2 catalog

The inventory has been completed, vendors have been contacted, and you have verified that the DB2 Version 7 sample databases and IVP jobs still exist or have been reinstalled, but your Version 8 tapes still haven't arrived (or maybe haven't even been ordered). In the meantime, you can clean up the DB2 catalog. If your shop has been using DB2 for a while, you might have accumulated 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. After you have cleansed the DB2 catalog, reorganize the catalog table spaces that you are allowed to reorganize in Version 7. Reorganizing the catalog is not only a good practice for the day that you migrate to enabling new-function mode (ENFM); it might also help out with catalog performance, and the elapsed time, of the ENFM migration step. ENFM is the Version 8 migration mode that converts most of the DB2 catalog to Unicode and

long names. The REORG utility converts the catalog table spaces from EBCDIC to Unicode. Having the catalog well organized before making the move to ENFM can improve the performance of the REORG utility. The REORG utility with the SHRLEVEL REFERENCE is available for all DB2 catalog table spaces during compatibility mode.

Migrate stored procedures managed by DB2

If you are still using stored procedures that are managed by DB2, now would be a great time to start migrating them to run IBM Workload Manager (WLM). You can make better use of stored procedures if they are managed by WLM, not to mention that DB2 Version 8 does not support stored procedures that are managed by DB2. So start Resource Recovery Services (RRS), set up WLM service classes, add the WLM ENVIRONMENT variable names to your procedures with the ALTER PROCEDURE SQL statement and begin testing. Not only might your stored procedures realize better performance under WLM, but you will also be able to take advantage of some procedure keywords that are available only when using WLM.

Stored procedures managed by DB2 will continue to function in Version 8. However, you can no longer run the ALTER PROCEDURE statement against them. If you attempt to alter an existing stored procedure after upgrading to Version 8 compatibility mode (CM), the procedure will automatically be converted to being WLM-managed. Likewise, after upgrading to Version 8 CM, any new stored procedures that you create will be WLM-managed. By migrating your stored procedures before you migrate from Version 7, you make the process easier and retain more control over it.

Bind plans and packages

If you have plans that have not been bound in the past decade or two, that is another issue you should start planning to address. It is 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 always take advantage of the benefits the optimizer is capable of giving them. Both strategies have merit. However, you

are usually going to come out ahead if you do the bind rather than avoiding the bind. If performance issues arise because of an access-path change, open an electronic problem management record (PMR) and so that IBM Support Center can help you solve the problem. Adding custom SQL to an application or avoiding the bind process simply delays the inevitable. If you have a database request module (DBRM) or plan created before DB2 V2.3, it is not going to work after you migrate 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 might still have issues. They might not be issues that prevent something from working successfully in Version 8, but issues just the same. Consider putting a plan in place now, before starting your migration to Version 8 compatibility mode, so that you can 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 Version 7 and packages bound in Version 8. You should also retain output from Version 7 or Version 8 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 access-path changes for your package. Other tools (such as Path Checker) can also help you determine where access-path changes have occurred.

If you are using packages, you might be confused by the references to not having bound a plan. Remember, these references pertain only to plans that have had DBRMs bound directly into them. If your plans use a PKLIST and access packages appropriately, then you only need to be concerned with binding those older packages.

PDSEs and load libraries

If you have been reading about DB2 Version 8, you might have come across a new term, partitioned data set extended (PDSE). It has been around for quite a while, although not as long as the other term that you are more familiar partitioned data set (PDS).

The description of the DSNALLOC installation job step in the *DB2 Version 8 Installation Guide*, GC18-7418, states that PDSE is required for the DB2 SDSNLOAD target library. A PDSE is the default for this job. However, it turns

out that PDSE is not required – yet. Nothing in Version 8 exceeds the 16 MB limit. So, for DB2 Version 8, there is no need to use a PDSE for distributed load modules. The requirement for PDSE is enforced for DB2 9, though, because all load modules in that version cannot be contained to less than 16 MB. If you are unfamiliar with PDSEs, have a discussion with your z/OS system folk about them. Many DB2 Version 8 shops have made the PDSE choice. Regardless of which format you chose to use, PDS or PDSE, make sure SMP/E knows which one you picked. And if you choose PDSE, be sure that PDSE service is current. There is more information about PDSE in the *DB2 Installation Guide* and in the *DB2 Version 8 Program Directory*, GI10-8566, both available from the DB2 library at **ibm.com**/software/data/db2/zos/Version 8books.html.

There is another reason you should consider using PDSEs. You might want to consider using PDSEs for managing your stored procedures in your stored-procedure address space. A PDSE allows the extent information to be dynamically updated. This might 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 PDSEs for stored procedures or target load libraries, do some homework first. Take some precautions if a PDSE is going to be shared outside of a sysplex or global resource serialization (GRS) 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 PDSEs yet, there is no problem using them (they have been around for years), and it's your choice in Version 8 (although they will be required in DB2 9).

What about compilers?

As soon as planning for a Version 8 migration begins, check the language you use for programming with DB2. Plan to run with current compilers and with a new version of DB2. DB2 no longer supports certain programming language

compilers. For example, consider COBOL. The Enterprise COBOL for z/OS and OS/390 Version 4 compiler (5655-G53) is supported by DB2 Version 8 and should be the complier you are using. However, DB2 Version 8 does work with IBM COBOL for OS/390 and VM Version 2 Release 2. Service support for COBOL V2R2 ended in December 2004, and support has been withdrawn for all other versions of the COBOL compiler that are out of service. If you want to take advantage of the integrated SQL coprocessor, you must use Enterprise COBOL V3.4, V3.3 or V3.2 (5655-G53) with APAR PQ83744. Service support for V3R2 ended in October 2005 and support for V3R3 ended in April 2007.

The following table summarizes all of the COBOL compilers and the date on which service is withdrawn.

COBOL compiler	Part #	Withdrawn from service	Runtime library supported
OS/VS COBOL	5740-CB1	Jun 1994	Language Environment only*
COBOL/370	5688-197	Sep1997	Language Environment only*
VS COBOL II	5668-958	Mar 2001	Language Environment only*
COBOL for MVS™ & VM V1R2	5688-197	Dec 2001	Language Environment only*
COBOL for OS/390 & VM V2	5648-A25 (MVS only)	Dec 2004	DB2 V8
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	N/A	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 IBM Language Environment® (LE) is utilized. The preceding table indicates which compilers require LE support. Compiling 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 new functions in DB2 Version 8. If there is no need to use a 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 can 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, make sure that you are using either IBM Enterprise PL/I for z/OS V3.4 or later (5655-H31). IBM PL/I for MVS and VM V1.1 (5688-235) is old, and end of service has been announced, but it works if you avoid using a new function. If you will be using the DB2 precompiler services, you must use the DB2 Coprocessor provided with Enterprise PL/I for z/OS V3.2 with the fix for APAR PQ84513 applied, or a later release of Enterprise PL/I for z/OS and OS/390. Enterprise PL/I for z/OS V3.2 is out of service. V3.3 goes out of service in September 2007. Of course, just like the COBOL compilers, there are caveats. Although the IBM PL/I for MVS and VM V1.1 (5688-235) compiler is still in service, it is old, so make plans to move to a more-current PL/I compiler. Version 2 and Version 3 Release 1 of the IBM Enterprise PL/I for z/OS and OS/390 (5655-H31) compiler are no longer in service. For the C and C++ programming languages, both the 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 Version 8.

Migration guides are available from the Web for PL/I (SC26-3118, at publibfp.boulder.ibm.com/epubs/pdf/ibm3m101.pdf) and COBOL the *Compiler and Run-Time Migration Guide*, (GC27-1409, at publibfp.boulder.ibm.com/epubs/pdf/igy3mg10.pdf). These should be your primary resources if you need additional details 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. You can download the program directory for DB2 Version 8 from the IBM DB2 library Web site, **ibm.com**/software/data/db2/zos/Version 8books.html.

Checking your compiler levels might 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 need 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 Version 8 requires. If the tools interface directly with DB2 in any way, make sure they can handle the new functionality in DB2 Version 8.

Prepare your team for the migration

It's never too soon to prepare your team in anticipation of the eventual migration to DB2 for z/OS Version 8. Start having sessions, classes, and meetings to discuss 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 provided by IBM. Also, read Roger Miller's article *"Greatest Hits and Myths: DB2 UDB for z/OS V8"*. Go to the DB2 Support Web page at **ibm.com**/software/data/db2/zos/support.html and search for the title or author. You might also like to download his presentation on planning migration from ftp://ftp.software.ibm.com/software/data/db2zos/SHAREdb2zv9MigrationMiller.pdf.

Version 8 effects on DSNZPARM

Before starting your migration, you need to understand how DB2 Version 8 affects the values you have coded in DSNZPARM macros. DB2 for z/OS Version 8 has made a significant number of changes to the DSNZPARM macros. Not only have the default values for quite a few Version 7 keywords been changed in Version 8, some of the ZPARM macro 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. You don't need to know the details of these changes until you at least have compatibility mode running. However, during the planning stages for your migration, it's worthwhile to look at a few of the ZPARM keywords that might have an effect on your use of DB2, while you have some time to determine if further action will be required. The more preplanning and clean-up you perform while still running Version 7, in preparation for upgrading to Version 8, the better off you will be.

Research the DSNZPARM keywords whose defaults have been changed

First, there are keywords whose defaults have been changed. If you haven't coded a value for one of these keywords, then the behavior you are used to in DB2 Version 7 could be a little different in Version 8 after the DSNZPARM (DSNZPARM is a data-only subsystem parameter load module that contains DB2's execution time parameters) is assembled and linked. Additionally, by reviewing the new default values being delivered in Version 8, you might get some hints or ideas for changes you can make to ZPARM today in Version 7 in preparation for Version 8. You do not necessarily need to change a keyword simply because the default changed. However, the default might have been changed for a reason worth further investigation. For example, some of the defaults were changed because DB2 will perform much better with updated values.

The defaults have been changed for at least 18 keywords; some of those changes are fairly insignificant. For additional information about all of the changes, see the Redbooks publication (available for download from the Web) *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know,…and More*, SG24-6079. A few keywords are worth mentioning here though, and they might catch your attention. For example, the defaults for the DSN6SYSP keywords CTHREAD, MAXDBAT, and CONDBAT have been changed from 70 to 200, 64 to 200, and 64 to 10,000, respectively. These are the other keywords on the DSN6SYSP macro whose defaults have been changed:

- *Log apply storage: The default for the LOGAPSTG keyword was changed from 0 to 100. This is a good example of a parameter that should always have been set. If you haven't set LOGAPSTG to 100 yet, you should. For more*

information, see the article about DSNZPARMs in the spring issue of zJournal Magazine *at* www.zjournal.com.

- *Maximum number of data sets that can be open at one time: The default for the DSMAX keyword has been increased from 3000 to 10,000.*
- *Checkpoint frequency: The default for the CHKFREQ keyword is now 500 000, instead of 50 000 as in Version 7.*
- *Extended security: The default for the EXTSEC keyword is now ON.*
- *Dynamic statement caching: The default for the CACHEDYN keyword is now ON. In addition, you can now use the SET SYSPARM command to change the CACHEDYN value. 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 the setting for this keyword.*

Investigate DSNZPARM keywords that have been removed

Certain DSNZPARM keywords have been removed from DB2 Version 8. Some of them you probably never realized were there in the first place; others could present some real problems if you have to change code after the keyword is no longer available.

PKGLDTOL is one of the more significant keywords being excluded from Version 8. This ZPARM keyword was delivered by fixes for APARs PQ59207 and PQ76444. With these PTFs applied, a plan or package is required for certain SQL statements at the local site. In Version 7, PKGLDTOL could be enabled (set to YES) to circumvent the DB2 requirement, allowing these SQL statements to work as they did in Version 6. PKGLDTOL is removed in Version 8, eliminating the option to override the DB2 requirement. For more information about this keyword, see the section on making sure that PKGLDTOL is turned off.

In Version 7, the default value for the INLISTP keyword on the DSN6SPRM macro is set to 0, or turned off. In DB2 for z/OS Version 8, INLISTP is set to 50, so it is turned on by default. Because this value is not on a panel, you might 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. For more information about this keyword, see the following APARs for DB2 Version 7:

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

A few more DSNZPARM keywords that existed in DB2 Version 7 but have been removed from DB2 Version 8 are OPTCCOS1, OPTCCOS2, and OPTSUBQ1, all on the DSN6SPRM macro. These keywords were provided via APAR in Version 7. Although these keywords no longer exist in Version 8, the functionality provided by these particular ZPARM keywords has been incorporated into the Version 8 base code. Minimum problems were reported in Version 7 by the functionality provided by these DSNZPARM keywords 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. Make sure to review the following APARs:

- *Both PQ50462 and PQ81790 introduced OPTSUBQ1 to fix a problem with the access-path selection logic to correctly add noncorrelated 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 these APARs will help you understand how these DSN6SPRM keywords can affect your Version 7 subsystem and what behavior changes will result from modifying them.

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

Make sure that the PKGLDTOL keyword is turned off — everywhere
Hopefully, this warning affects only a small handful of customers who are upgrading to DB2 Version 8. This ZPARM keyword was mentioned briefly elsewhere in this paper, but it deserves its own section.

As part of your Version 8 upgrade planning, check the source for all of your DSNZPARM members (Development, Test, QA, Production, and so on) for the PKGLDTOL keyword on the DSN6SPRM macro. If PKGLDTOL is coded, make sure it is set to NO. If you can't find it anywhere (it hasn't been coded), that's good because the default is NO. This is one parameter that should not be set to YES anywhere in your Version 7 subsystem.

When was this keyword introduced? Back in DB2 Version 6, you could use a few selected SQL statements in an application without the use of a plan or package, and without performing a bind. 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*

This became a problem in DB2 Version 7 when the oversight was "corrected," based on the incorrect assumption that no one knew about the loophole or had taken advantage of it. For Version 7 only, a new DSNZPARM keyword (PKGLDTOL) was provided to allow those customers who did take advantage of the "feature" a way of successfully migrating to Version 7 and time to put the necessary packages in place. Specifically, PKGLDTOL could be enabled (set to YES) to circumvent the behavior in Version 7, allowing these SQL statements to behave as they did in Version 6. PKGLDTOL is removed in Version 8, thus eliminating any option to override the DB2 requirement for these SQL statements.

Do yourself a favor, and verify that PKGLDTOL is not turned on in your subsystems. If it is, your next step is to determine if any of the SQL statements mentioned previously are referenced in an application without the use of a plan or package. After you find the applications, fix them so you will not have any hidden inhibitors to a successful Version 8 migration.

To read more about this keyword, see the information about defining DB2 initialization parameters in the *DB2 Installation Guide*, GC26-9936-04. Make sure you are looking at the most recent version of the manual that is available. You can also look up APARs PQ59207 and PQ76444 on the IBM DB2 Support Web page.

Become familiar with the CLAIMTBA keyword

You should also know about the CLAIMTBA keyword, which was added to Version 7 by an APAR. The ZPARM switch added in this fix, if set on, causes DB2 to always acquire claims on table spaces (and partitions) before acquiring any claims on any indexes. The switch 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.

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, allowing current behavior to remain the same until you take some action. Setting CLAIMTBA to YES forces the new behavior. Version 8 accesses data first, and then the indexes.

This keyword will not cause you any issues if you ignore it. However, the processing for the keyword was built into Version 8 and the ZPARM was removed. By taking advantage of that processing while you are still running Version 7, you can become accustomed to its behavior and avoid any potential surprises later on.

If you decide that the fix for APAR PQ96628 is of interest to you, the PTF is part of RSU 0506. You should also apply the following two APARs along with PQ96628:

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

Use of DFSORT by DB2 Version 8 utilities

If you use the IBM DB2 utilities and any sort product not from IBM, it is important for you to understand the IBM DFSORT™ product before using the IBM DB2 utilities. If you do not use the IBM DB2 utilities or if DFSORT is your sort product, you can skip this section.

DB2 for z/OS Version 8 and its use of DFSORT is periodically a topic of discussion on International DB2 User's Group DB2 Database Discussion list (DB2-L listserve, available at www.idugdb2-l.org/archives/db2-l.html). Here is a summary of what you need to know about this subject.

First and foremost, the IBM DB2 for z/OS Version 8 utilities require DFSORT. This support was delivered by the PTF for APAR PQ68263 (PTF UQ90054) on RSU0312. In fact, with the delivery of the fix for APAR PK04076 (UK03983), message DSNU1640I will be issued by DB2 when the IBM DB2 utilities are unable to locate DFSORT code at the APAR PQ68263 maintenance level. However, there is good news.

You already have DFSORT installed. You might not be using it, but it is there. DFSORT is always shipped and installed with z/OS. No license is required for the DB2 utilities to use DFSORT. The only action you need to take is to add the DFSORT load libraries to link list after the original equipment manufacturer (OEM) sort libraries, or add a STEPLIB or JOBLIB statement to the batch JCL for DB2 utilities. DB2 for z/OS Version 8 has a license to use DFSORT, so you do not need a license for any external sort product. However, no task outside of DB2 can use DFSORT without being licensed. Use of DFSORT code by products other than DB2 requires a license.

Maintenance for DFSORT is also shipped with z/OS. If you encounter problems with the DB2 usage of DFSORT, open an ETR with IBM Support Center, just as you do today for any other issue related to DB2.

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

- DB2 Version 8 Installation Guide, *GC18-7418, available with all of the DB2 Version 8 manuals at* **ibm.com***/software/data/db2/zos/Version 8books.html*
- DB2 UDB for z/OS Version 8: Everything You Wanted to Know, … and More, *SG24-6079*

The DB2 Version 8 utilities use only DFSORT. You have no other choice. Not much was published on the subject before APAR PQ68263. Some customers thought this was just a suggestion, because some utilities seemed to work just fine with sort products other than DFSORT. Then some customers started to get a message that the utility they were running could not find certain DFSORT modules. As a result of the questions that started to arise, APAR PK04076 (PTF UK03983) was published, enabling a new error message. Now, if you have applied the fix for APAR PQ68263 (the maintenance that enabled the exclusive use of DFSORT by the Version 8 utilities), message DSNU1640I is issued when a utility is unable to locate the DFSORT code.

Having DB2 for z/OS use DFSORT allows DB2 utilities to test and support DB2 more thoroughly. For example, in Version 8, the utilities allow sorting for 32K pages, which works with DFSORT. DB2 users can benefit now that the utilities take advantage of DFSORT. Like DB2, DFSORT is an IBM product. That gives the DB2 team the opportunity to know more about DFSORT than about other vendor's sort products in use today. And because the DB2 Version 8 utilities use DFSORT exclusively, the utilities have the opportunity to take advantage of functions and features in DFSORT that could not be considered in the past. The focus on a single sort utility eliminates the need to ensure that a given sort feature or function used by a DB2 utility also works with sort products other than DFSORT.

Commonly asked questions about DFSORT and DB2

So what are some of the questions people ask about DFSORT and more importantly, what are the answers?

"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, without a license, you cannot use DFSORT outside of the DB2 utilities.

"Do all the DB2 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.

"How does DB2 know to use DFSORT and not my other sort product?"

This question has a few answers. It is your responsibility to make sure that the load libraries are in the right place and in the correct sequence. In the search list, the two DFSORT libraries must *follow* the libraries for other sort products that are in use. For example, if you are using SyncSort and running from the LINKLIST utility, make sure the SyncSort libraries are first, followed by the DFSORT libraries. You can also use STEPLIB or JOBLIB to access the DFSORT libraries, but that can be 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 uses the same names as 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.

"Do the DFSORT load libraries have to be authorized?"

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

Informational APARs for DB2 Version 8 and DFSORT

Before you are well into your migration, make sure to read the three informational APARs that describe how the DB2 utilities in Version 8 use DFSORT. Bookmark these APARs on the Web, so you can make them your first stop when you have questions or issues about how DB2 is interacting with DFSORT. The informational APARs, II14047, II14213, and II13495, are available from the IBM Web site at:

ibm.com/support/docview.wss?rs=0&uid=isg1II14047

ibm.com/support/docview.wss?uid=isg1II14213

ibm.com/support/docview.wss?uid=isg1II13495

Reserved words for DB2 Version 8

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 message DSNH083I) unless they are a delimited identifier (marked off by quotation marks ["], for example). And why might reserved words be a problem? When they are used out of context, DB2 might confuse them with DB2 keywords. To complicate matters, DB2 allows many keywords to be used 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 Version 8, you might discover that you have been using a reserved word from Version 6, but the context has not caused a problem until now.

To avoid problems with reserved words, don't use them. There is a table of DB2 reserved words in the *SQL Reference*, SC18-7426. Make sure you have the most current edition of the manual, to avoid missing a reserved word because your documentation was out of date.

To help you start thinking about these reserved words, here is a list of reserved words added in DB2 Version 8, current as of February 2006. Check this list carefully if you are running Version 7 and planning an upgrade to Version 8. This list contains a few words that could trip you up. To be more careful, check the list of reserved words in *SQL Reference for Cross-Platform Development*, which includes words for the DB2 family (as in the SQL Reference) and also the list of standard reserved words. This document is available from **ibm.com**/developerworks/db2/library/techarticle/0206sqlref/0206sqlref.html.

ASENSITIVE	NONE	ROWSET
ENCRYPTION	PADDED	SECQTY
ENDING	PARTITION	SECURITY
EXCEPTION	PARTITIONED	SEQUENCE
HOLD	PARTITIONING	SIGNAL
INCLUSIVE	PATH PIECESIZE	SUMMARY
ITERATE	PREVVAL	VALUE
MAINTAINED	QUERY	VARIABLE
MATERIALIZED	REFRESH	VOLATILE
NEXTVAL	RESIGNAL	XMLELEMENT

SYSOPR

DB2 can catch you in its web, 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 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." Not the DB2 privilege set SYSOPR – this SYSOPR is just another authid. So why did the command work? This is where the web gets tangled. When you install DB2 Version 7, the default setting in DSNZPARM for the installation SYSOPR authid is SYSOPR. If you don't change it (and many users don't), that is what it remains. And leaving SYSOPR as the default lets you enter the command.

This action changes in Version 8. If you are security conscious and concerned about who can run DB2 commands such as -STOP DB2, this change is good news. In DB2 Version 8, the authorization for the user ID 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 distribute this authority widely in Version 7 and just used the ZPARM default, you are going to have a lot fewer DB2 commands being issued from SDSF or from the console in Version 8. You should resolve this authorization issue before you upgrade to Version 8 so you don't have to worry about processes no longer working after your upgrade is complete.

Some additional good news in Version 8: authority for commands can be given to a secondary authority ID. This can make it easier to set up the right people to have this privilege now that command access has to be explicitly granted. In addition, you can use the SET SYSPARM command to change the installation SYSOPR value. The only catch to doing this is that you must be the installation SYSADM when you issue the SET command. If you are not, all other DSNZPARM changes take affect except the change to installation SYSOPR.

Excessive CPU consumption

This section discusses CPU consumption by DB2 for z/OS Version 8: what's true and what is a myth. For example, what is true is that any new version of DB2, including Version 8, will probably use more CPU resource than the previous version. However, Version 8 brings some other "challenges." Although-64 bit processing enables you to do more with storage, it also uses instructions and addresses that can be up to 50 percent larger than in the past. Larger instructions and addresses mean bigger registers and control blocks. In some situations, this can all translate to additional consumption of CPU resource when the same Version 7 workload is run in Version 8. How much of an increase you experience depends on the type of workload you are running. Some shops have seen no (and in some cases a negative) increase in CPU utilization.

Bind CPU cost, especially for dynamic prepares, might go up if you use the RUNSTATS frequency value option, and if the FREQVAL COUNT in SYSCOLDIST is greater than 100 because Bind removes duplicates. If this is happening to you, see APAR PQ94147.

You can realize CPU savings thanks to multi-row FETCH, discussed later in the overview of new function mode. An additional benefit is that you don't have to wait until new function mode (NFM). You can code 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), have an application use the limited-block fetch function in Distributed Relational Database Architecture™ (DRDA)® and you can automatically use multi-row FETCH and receive immediate CPU savings.

And sometimes you see what looks like high CPU consumption, but it turns out not to be the case. For more information about that situation, see APAR PK25427 for DB2 and the related z/OS APAR OA15666 that change 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 impact on real storage. Page-fixing the buffer pools can reduce your CPU usage just like eliminating I/O can decrease your CPU usage. You eliminate I/O by keeping more in the pools; 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. Compression is something you should take 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. Consider turning on compression for more table spaces.

Another myth 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 worthwhile. But it was discovered that this enhancement didn't provide the benefit that was expected, especially considering the additional expense it added. APAR PK21237 came out in March 2006 (the related PTF came out in May 2006) that reduced the number of write engines to 300 again. This immediately buys you back, on average, about 100 MB of storage without any effect on your performance. This APAR also fixes some issues related to releasing stack storage and combining engines to a single pool.

Final comment: virtual equals real. Much discussion of Version 8 suggests increasing pool sizes. However, even though you are in a 64-bit environment, storage is not unlimited. You can use only what you have backed completely with real storage.

Plans, packages and DBDs in DB2 Version 8

Rebind after upgrading to DB2 Version 8

After migrating to DB2 for z/OS Version 8, you might notice a decrease in performance for some queries – specifically, those long-running applications with SELECTs and FETCHs that process lots of rows and return lots of columns. Those queries ran just fine in DB2 Version 7 and didn't seem to have problems until your Version 8 migration. What can you do to improve the performance of these queries?

First, do you have all of your EXPLAIN data for this application from your Version 7 environment? Did you also save the Version 7 accounting data for this application so you have evidence of how it might have actually run in Version 7? If you have checked the EXPLAIN data you captured for this application after migrating to Version 8 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 revelations, a possible solution might be as simple as a rebind.

First, some background to help you understand why rebinding might be the right solution. You have probably heard advice about not using SELECT * or about minimizing the number of columns you specify on a SELECT. That advice is based on the fact that DB2 moves one column at a time between the DBM1 address space and the application address space. The more columns you specify on a SELECT, especially columns you have no intention of ever referencing, the more it costs to process 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?

DB2 Version 3 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 processed repeatedly, and then 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 can be.

How is this related Version 8? If a plan or package is using an SPROC in Version 7, the SPROC is using 31-bit code. When you run that same plan or package in Version 8 without rebinding it first, Version 8 expects it to use 64-bit code, which it doesn't. As a result, DB2 disables the procedure. The only way you can reenable the procedure is by rebinding it. Until you do that rebind, and if the plan or package uses an SPROC, the performance of your application will be degraded. Do the rebind, and you should see a performance improvement.

You can do the rebind in compatibility mode (CM) or in new function mode (NFM). If you rebind the plan or package in CM, there is no reason to bind it again in NFM because of the SPROC. After moving to Version 8 NFM, if you adjust some of the tables and indexes for better performance, you'll need to rebind to get further performance improvements.

Suggestion: If you attempt a rebind in Version 8 and it is not successful, check whether you can rebind that same package or plan in Version 7 before contacting DB2 Support for help. This precaution can avoid trying to debug a problem that does not exist.

After you migrate to Version 8, it's important to leave behind any reluctance to bind. 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 any packages and plans that have SQL if they might be able to take advantage of index-only access from an index being changed to nonpadded.

For a discussion of other optimizer enhancements that are available in new function mode (NFM) if a plan or package was already bound while in compatibility mode (CM), see the overview of NFM later in this paper.

Bottom line: When defining your migration plan, you need to include a rebind strategy for all of your packages (and plans, if you have plans with DBRMs bound into them directly) after you complete your move to NFM. Rebinding ensures that all packages (and plans) are in the correct format for your new catalog.

Clean up the SPT01 package

One possible side effect of binding a package in Version 8 is an increase in size for the SPT01 package table in the directory. SPT01 is a directory table space used to store skeleton package tables (SKPTs). It cannot exceed 64 GB. Before starting your upgrade to Version 8, or even while you are in compatibility mode (CM) but before doing any binds, check the size of SPT01. If it is in the 30 - 40 GB range (or greater), you might not have enough room to rebind all of your packages in Version 8, and you might run into problems during CM. Rebinding (and binding) a Version 7 package in Version 8 can significantly increase the package size. Remember, in Version 8 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 percent larger.

How do you address this potential issue? Start cleaning up your packages. If you use packages and have a fairly large SPT01, check for excess package versions. If you are not using package versions, determine whether you have packages that are no longer in use that you can free up. Whatever you do, stay below the maximum size for the SPT01 file page set.

How do you make a DB2 DBD dizzy?

The short, cute answer would be to migrate to Version 8. Just think: first you're Version 7 and then you're Version 8 and then you're Version 7 and then you're Version 8 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.

The question had to do with expanding database descriptors (DBDs) in Version 8. Here is the answer:

- Version 8 requires that the CCSID information in the DBD be accurate. CCSID information in DBDs written prior to Version 8 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 Version 8, 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, because it isn't free. If such a DBD is later altered and written out by Version 7, 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 Version 8.

- When we read a DBD written prior to Version 8, we transform it to conform to Version 8 format.

- When we write a DBD in CM, we transform it into Version 7 format.

When you take those things together, you can see that we might end up with a DBD in Version 7 format with the CCSIDs corrected that would stay that way, without some form of alter to it in ENFM or NFM. Issuing a -DISPLAY DATABASE(*) command in ENFM or NFM causes all the DBDs not yet loaded in Version 8 to have their CCSID info corrected and written out in Version 8 format. But, these DBDs are probably less interesting than those used in CM, which would already have been written out in Version 7 format. Because this info is in the directory, there is no query you can use to find out which is which. The cost of transforming a DBD from Version 7 format to Version 8 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 suggest making some small change to the DBDs you depend on in NFM. 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. Alternatively, you can alter them to use our sliding secondary so that you don't have to worry about primary or secondary any more.

EDMPOOL

While you're thinking about packages, consider the EDMPOOL. The Version 7 EDMPOOL has been broken into three 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. Avoid your impulse to decrease the size of the EDMPOOL because there is less in it. Instead, leave it alone, for at least a little while. Plans and packages will be larger in Version 8 after they are rebound. Wait until you have completed all of your rebinds and have had a chance to determine the actual EDMPOOL utilization before reducing its size.

DESCRIBE WITH STATIC option

The default for the DESCRIBE WITH STATIC option on installation panel DSNTIP4 (DSNZPARM parameter DESCSTAT on macro DSN6SPRM) has been changed to YES in Version 8. This can lead to an increase in package size, increasing the storage requirements of SCT01 and possibly the EDMPOOL below the bar.

Actions to take now to avoid bind-related surprises in Version 8 NFM

Certain processing that occurs during enabling new-function mode (ENFM) can cause unpleasant surprises when you move to new function mode – specifically, the automatic rebinding of invalidated packages. How does a package become invalidated? If you drop an object that a package is dependent on, such as a table, the package is no longer valid. You probably don't plan on dropping any objects during your move to NFM, but DB2 does.

During enabling new function mode (ENFM), DB2 converts most (but not all) of the catalog tables to Unicode. However, SYSCOPY does not get converted, and neither does SYSDUMMY1. SYSIBM.SYSCOPY resides in its own table space (DSNDB01.lSYSCOPY), so it isn't an issue. However, SYSIBM.SYSDUMMY1 does not. It is in DSNDB06.SYSSTR with 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. And how do you move a table to a different table space? You drop it and recreate it.

To avoid being surprised by automatic rebinds, take action before starting you migration. First, search SYSPACKDEP and find any packages that have a dependency on SYSDUMMY1. Then build a rebind job that includes all of those packages, and run that job at the completion of the REORGs that convert the catalog to Unicode. That should put you back in control.

DB2 Version 8 and Unicode

For some reason there is some confusion on the subject of Unicode. 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 the attention of 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 mentioned previously, during ENFM, the DB2 catalog is converted from EBCDIC to Unicode. There are also other areas of DB2 where Unicode plays a significant role. To help relieve the confusion surrounding Unicode and DB2, this section provides a brief description of what Unicode is, discusses why Unicode might cause you some concerns, and points you to resources that can help you learn more about Unicode.

What is Unicode?

First, you need to know some terms that you will 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 uppercase character "A", and the hex value "81" represents a lowercase character "a". These hex representations of your data are called code points. When you create a group of these code points, 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), you have probably run into code page issues. Keep in mind that code pages are not the same as coded character set identifiers (CCSIDs). A *CCSID* is just a numeric value that *identifies* a code page. CCSIDs are part of the metadata that describes the data in a DB2 subsystem. For example, if you are using a US English code page, it is represented by CCSID 037. There is also a special-case US English code page (represented by CCSID 1140) that includes

the euro symbol. Both CCSIDs 037 and 1140 will probably be valid for most U. S. shops. A CCSID in DB2 is also specified for the entire DB2 subsystem and once specified, should not be modified. If you have modified any CCSIDs after specifying them, you might need to address that problem when you start planning for your migration to DB2 Version 8.

But what is Unicode? The official definition from the Unicode Web site (www.unicode.org): "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." Another quote from the Unicode Web site sums up why Unicode is important: "Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language." Everyone gets to work with the same character representation. 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 code 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?

Whether you should be concerned about Unicode depends on the state of your CCSIDs. While you are still on DB2 Version 7 and planning for your Version 8 upgrade, it is imperative to ensure that the CCSID values in your DB2 Version 7 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.

Validate the CCSID that is set up

To find out if you should be concerned, first check DSNHDECP to validate what CCSID you currently have DB2 set up to use. For shops 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 CCSSID 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 might 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	Description
SCCSID	Single-byte EBCDIC
MCCSID	Mixed EBCDIC
GCCSID	Graphic EBCDIC
ASCCSID	Single-byte ASCII
AMCCSID	Mixed ASCII
AGCCSID	Graphic ACSII
USCCSID	Single-byte Unicode
UMCCSID	Mixed Unicode
UGCCSID	Graphic Unicode

How do you know whether you have a Unicode concern? You cannot make a determination by simply running a query against the DB2 catalog. Instead, you must perform two tasks. First, run job DSNTIJP8. This job is delivered via APAR PQ84421 (PTF UQ85439 on RSU 0406). It is the DB2 Version 8 flavor of DSNTIJPM. Do not run DSNTIJPM delivered with DB2 Version 7, which is the verification job for migrating from Version 6 to Version 7. The Version 8 flavor

of DSNTIJPM is delivered on the Version 8 distribution tapes. This job should be run on DB2 Version 7 at your earliest convenience to identify any concerns that need to be resolved before commencing a migration to Version 8. Any CCSID issues should be resolved before you start your migration to DB2 Version 8 compatibility mode.

Open an ETR for CCSID 500

If you discover you might not be using the CCSID you should be using, do not assume that you have to change it. After running DSNTIJP8, if you have CCSID concerns, your second task is to open an ETR as soon as possible with IBM Support Center, specifying DB2 "CCSID 500 prior to migrating to Version 8" as the subject line. Level 2 can 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 IBM Support Center. 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, if changes are necessary.

When you specify a CCSID that does not match the data, the possible result is the improper translation of some characters. Here is an example. A piece of data is encoded using CCSID 500 although DB2 Version 7 is not enforcing that CCSID and the data seems to be using CCSID 037. When the Version 8 migration is complete, the DB2 catalog will be encoded using Unicode. If you were using a 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 CCSID. Now the character returned will be a right corner. A character-translation error has just occurred. Using CCSID 037 and 500 as examples and because CCSID 500 seems to be the CCSID miscoded in most cases, there are actually seven characters that have different hex values, as shown 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 some data is unaffected by Unicode. For example, any column defined as "FOR BIT DATA" is not affected, nor is numeric data stored as binary, packed, or floating.

One concern often mentioned in a first-time Unicode discussion is the amount of space the catalog will take after encoding it in Unicode. When the catalog is converted to Unicode, how much the catalog might increase in size depends on the type of data being converted in the catalog. For example, if your current CCSID is 037, 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 many special characters or double-byte characters, the Unicode equivalents will probably use more space than the EBCDIC character.

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

What else could be affected by Unicode? 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. After you complete your upgrade of Version 8 to new function mode (NFM) and NEWFUN(YES), all of your DBRMs are created using Unicode. And all SQL that is stored in SYSIBM.SYSSTMT and SYSIBM.SYSPACKSTMT is stored in Unicode. Literals might be in Unicode, the results from an SQL statement might be Unicode, and SQL included in instrumentation-facility, component-identifiers (IFCIDs) might be in Unicode (depending on the ZPARM value that controls the format).

Unicode and IFCIDs

After you have completed your migration to DB2 for z/OS Version 8, your catalog will have been converted to Unicode. With that change, any SQL information added to certain IFCIDs might also be formatted in Unicode.

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 Version 8) on panel DSNTIPB. Your choices are NO (the default) and YES. If you specify NO, information added to the IFCIDs remains in EBCDIC format as in previous versions of DB2. If YES is specified, some of the information added to IFCIDs will be in Unicode. NO allows the information to remain in EBCDIC format, and therefore making it easier to work 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 might seem slight, it is still a cost. With the number of IFCIDs that some organizations cut, it's something to think about.

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 good tool should be able to handle Unicode in a DB2 Version 8 environment.

Unicode resources

Here is a list of resources to help you learn more about Unicode:

- *There are two must-read informational APARs containing all of the latest 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).*
- *Read APAR PQ84421: Add DSNTIJP8 - Version 8 Pre Migration Checkout Job - TO Version 7. This APAR, available for DB2 Version 7, allows you to check your DB2 catalog to identify potential Unicode problems.*
- *See the presentation by Chris Crone, "Unicode and DB2 Version 8, What You Need to Know," on the DB2 support site.*
- *For more white papers and presentation, and similar documentation about Unicode, go to the IBM DB2 support Web page,* **ibm.com**/software/data/db2/zos/support.html *and do a search on Unicode. You will get pages of Redbooks publications, presentations, white papers, articles, and a list of current APARs for and about Unicode.*
- *See the Unicode Web site at* www.unicode.org *if you are interested in a description not written by someone at IBM.*

Planning wrap-up

When you are doing your migration planning, make sure you have the most current copies of the documentation needed to perform the migration. Take the time to download a new, up-to-date set of manuals from the DB2 Web site,

ibm.com/software/data/db2/zos/Version 8books.html

Also, make sure you have access to the right books for the tasks 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 IBM Resource Access Control Facility (RACF®) software, you will need the *RACF Access Control Module Guide*. If you use Java™ software, you need to have Java manuals for that portion of the migration and 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.

Thinking of z/OS, remember to confirm that Unicode service is installed (and is installed correctly), has all of the required PTFs and is ready for DB2. DB2 Version 8 will not start if Unicode services are not configured correctly.

Hopefully, you are well on your way to validating that all required DB2 Version 7 maintenance has been applied. You must complete this step 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.

Tips: These are the three most important things to remember about migrating to Version 8: get some education; read the documentation (don't assume anything from previous migrations); and take it all very slowly, one step at a time, checking everything you complete.

The secrets to a successful upgrade to DB2 Version 8

It should come as no surprise that the two secrets to successfully upgrading to DB2 Version 8 are:

- *Planning*
- *Maintenance*

Although the benefits of planning are widely accepted, some shops are less inclined to see the value of keeping current on maintenance. Yet applying maintenance to DB2 on a regular basis can be as easy to accomplish, and as rewarding, as planning.

Reported customer usage for DB2 Version 8.

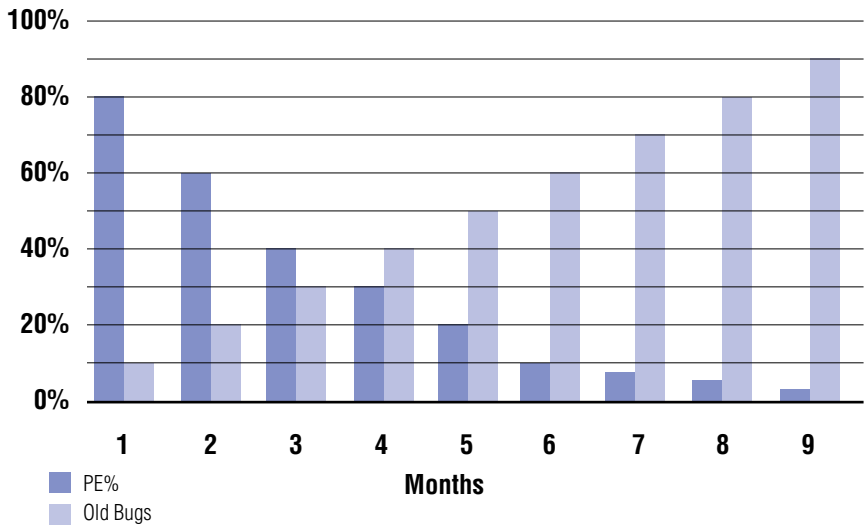

Figure 1.

The decreasing dark-colored bars represent the potential number of program temporary fixes (PTFs) that could be in error, in month order, by the month that the PTF was delivered. The increasing light-colored bars are the number of reported DB2 problems that already have a PTF available to fix the problem. Notice what happens around months 5 and 6. The number of problems that could have been avoided if maintenance had been applied far exceeds the num-

ber of potential problems that might have occurred if the maintenance was applied. Every time you hit a problem in month 5 or later, it's an outage that can be avoided if DB2 maintenance is applied in a timely fashion. As Figure 1 shows, you are better off putting maintenance on than you are avoiding it. Try to stay about 5 to 6 months current on PTFs, and apply as many hiper PTFs on a weekly basis (or what ever frequency your schedule can afford) to keep your DB2 subsystem up and running. It can be frustrating to have an outage that requires you to open a PMR and ship documentation to IBM to figure the problem out. But it can be just as frustrating 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.

No discussion of maintenance is complete without mentioning Consolidated Service Test (CST) and Recommended Service Upgrade (RSU), which together can be termed preventative service. By applying a few PTFs, you might save yourself potential frustration and possibly avoid having to deal with a needless ETR. 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 the fixes have been tested for your z/OS environment (including cross-product testing during CST, a customer-like sysplex environment, using industry representative workloads). Because an RSU tape contains a quarter's worth of tested PTFs, by waiting one additional quarter, all the PTFs fall into the 5+ month range mentioned previously. For details about CST and RSU maintenance, see **ibm.com**/servers/eserver/zseries/zos/servicetst/mission.html.

Need more encouragement to keep current with maintenance? With SMP/E V3.4, you can take advantage of 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 strategy for upgrading to DB2 Version 8?

An upgrade strategy for getting from Version 7 to Version 8 new-function mode can be as personal as how you fix your bagel in the morning. As long as you understand the rationale underlying the upgrade strategy you chose, you can be successful. This section explains one approach, to help you when you start to build your strategy or plan.

Step one: You need a plan. It can be a good plan or something bizarre, as long as it is something another person can follow and comment on, something that reveals that you have given some thought on how to get from point A to point B. It can be anything from a simple Microsoft® Word document or Excel spreadsheet to a full-blown work plan using a project management tool. Just write something down.

Step two: Use your 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 improvising, hoping everything will work out.

Step three: This is where we discuss one possible approach to moving through the three modes and complete migration to a full-blown DB2 for z/OS Version 8 subsystem. Use it as a catalyst for a team discussion of your potential migration plan. It is a starting point, to help you decide what will work best in your situation.

A strategy for compatibility mode

First stop: compatibility mode (CM). You have installed DB2 for z/OS Version 8 and you are now ready to start testing to see whether the installation was successful. 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 that the base DB2 code works correctly. Remember, the DBM1 and internal resource lock manager (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. Although different features are made available during the various migration modes, it's all still the same piece of code. So, put the code through its paces, and do want ever you can think of in an effort to break it. This is the mode where you should spend the most time. If you like to run IVPs, go ahead, but remember to use the Version 7 IVP while in CM. The Version 8 IVP will not work.

Make sure you run CM through at least one major event – end of month, end of quarter, or end of year, whatever event runs the majority of your most significant applications.

You are also getting most of the optimization improvements while you are in CM mode, so make sure to test those in CM. If you need to bind a plan or package to complete this testing, do it now. Don't wait until ENFM or NFM to find out you have an application that will not perform, because after you move to ENFM or NFM, falling back to CM or Version 7 are no longer options. You are in a "go forward" strategy.

And in CM you need to run all of your old SQL to see that it runs the same in Version 8 as it did in Version 7. You need to compare EXPLAIN data from Version 8 and Version 7, as well as check accounting reports from Version 8 and Version 7. This is the only way you can tell whether there is a performance issue. Telling your users you are now in Version 8 and asking them how things are running is not an accurate or an adequate measurement. The mere fact they know you changed something might skew their judgment.

In short, while you are in Version 8 CM, run everything you can find that ran in Version 7, and test everything you can locate. To the best of your ability, make sure you have no Version 8 issues of any kind while you are still in CM. It is the only time you can easily fall back to Version 7.

CM is also the time for an 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 a term used to describe a way of moving parts of your data-sharing environment forward while leaving other parts behind. Migrate a few members one weekend, migrate the rest the next weekend.

A strategy for enabling new function mode

Enabling new function mode (ENFM) converts the DB2 catalog to Unicode, long names and padded indexes. That is basically all that happens – ENFM is just CM with a new catalog structure. When you make the decision to go from CM to ENFM, you have already determined that Version 8 works to your expectations (excluding new features), so there is no need to spend much time in ENFM. Convert the catalog, validate that the conversion was successful, and proceed
to NFM.

A strategy for new function mode

The final step; new function mode (NFM). You have tested all you can test in CM; now it's time to find out how all of the new features work. In NFM, you are setting some bits in the directory and changing NEWFUN to YES in DECP, making the new features in Version 8 available for all to take advantage of. Consider NFM as the ultimate go-forward mode. If you have a problem while in NFM, you open a PMR and get it fixed. However, if you decide not to use the new features of Version 8, you can return to ENFM and disable all of those new NFM features.

In summary: Give a lot of consideration to the order in which you move your systems to NFM. You might want to migrate them backwards, meaning move your production subsystem to NFM first, and then migrate your other systems. Why? You need 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.

Overview of compatibility mode

You have completed all of the suggested planning, and the tapes have finally arrived or you have completed your downloads. It is finally time to start your migration from DB2 Version 7 to Version 8. Despite the importance of this new version, the most important thing to remember now is that this is 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. Thousands of successful migrations have been
completed already.

Compatibility mode (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 a 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 how DB2 uses 64-bit processing. When you are in CM you are ensuring that your DB2 functions correctly in this environment that is new to DB2.

Presentations often state that very little new function is delivered in CM other than 64-bit processing. That is not entirely true. The following list identifies some of the key items of DB2 Version 8 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. Do not assume that any of the items in the following list are 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 z/OS V1.3 (now out of service) or later is a requirement for a Version 8 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 Version 8 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.*
- *The 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 above 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 might 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 might be longer when represented in UNICODE, which can result in some string constants exceeding the maximum length for a string constant.*
- *The PGFIX(YES) option on the ALTER BUFFERPOOL command is available.*
- *IRLM PC=YES enforced. PC=NO is ignored in Version 8.*
- *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.*

CM coexistence

For customers who are migrating their data-sharing group to Version 8, you cannot have a discussion about compatibility mode without talking about coexistence.

Coexistence is during migration, the period of time in which two releases exist in the same data sharing group.

Many shops today take advantage of data sharing in DB2. When you migrate to a new version of DB2, one of the advantages 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 by allowing the group to stay active during the migration. 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. With Version 8, the ability to have two versions coexist is allowed only in compatibility mode.

You must follow a process to effectively migrate the members of a data-sharing group to Version 8. Ask yourself these questions:

First: Have you applied the fallback toleration fix for APAR (PQ48486) to all members of your data-sharing group? After successfully applying this PTF, you must restart all of the members of your data-sharing group while still on Version 7. Being able to fall back to Version 7 after migrating to Version 8 compatibility mode is a critical part of the Version 8 migration process. Applying this one PTF both enables fallback and allows you to run some members of a group on Version 7 while other members of that same group are on Version 8. With 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 Version 7 to Version 8 compatibility mode. Just remember, the toleration PTF must be applied and running on all members *before* you can migrate any member to Version 8.

Scheduling note: This can be a rolling outage, so that users do not see any down time. Schedule an appropriate time frame to incur this outage at the onset of your Version 8 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 Version 8 migration. While you are scheduling, you need to install the early or ERLY code changes on every z/OS image, and this step requires an IPL. Check for the service levels needed.

Second: Are you using DB2 Connect™ and the connection concentrator option with your data-sharing group? If so, you must verify whether the related maintenance has been applied. To take advantage of DB2 Connect Server connection concentrator while running in coexistence. DB2 Connect Version 8 Fix Pack 10, at a minimum, must be installed. You should be in Version 8 of DB2 Connect regardless, because DB2 Connect Version 7 is out of service. Also, 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 Version 7 and Version 8 running in coexistence in a data-sharing environment, you must also apply the fix for APAR PK05198.

Before continuing, here are some additional comments regarding DB2 Connect. It is true that the documentation states DB2 Version 8 will work with DB2 Connect Version 7.2 with Fix Pack 10, 11 or 12 applied, even though DB2 Connect Version 7.2 went out of service in September 2004. And a service extension was given to customers moving to DB2 Version 8. However, most of the best improvements in DB2 Connect were delivered with Version 8.1 and Fix Pack 4. But even that release level is probably not where you should be. If you are not planning on using connection concentrator, you need to consider installing, at a minimum, DB2 Connect Version 8.1 with Fix Pack 5. If you have any plans to use DRDA encryption support, you will need DB2 Connect Version 8.2 Fix Pack 7A to satisfy the minimum requirement, but again, you should consider Fix Pack 8.

Here is a summary of DB2 Connect versions and service levels and when they are appropriate:

- *DB2 Connect 8.2 Fix Pack 13 provides current service.*
- *DB2 Connect 8.2 Fix Pack 10 is required if you are using connection concentrator and data-sharing coexistence.*
- *DB2 Connect 7.2 Fix Pack 10, 11 or 12 satisfies the minimum requirement.*
- *DB2 Connect 7.2 is out of service. Although an extension was given to customers moving to DB2 Version 8, that extension has expired.*
- *DB2 Connect 8.1 Fix Pack 5 should be considered your minimum entry level.*
- *DB2 Connect 8.2 Fix Pack 8 is required for DRDA encryption support.*
- *Consider moving to the most current level of DB2 Connect, DB2 Connect V9 Fix Pack 2.*

Everything you need to know about DB2 Connect can be found at **ibm.com**/software/data/db2/db2connect/.

You can find all of the DB2 Connect Fix Packs at **ibm.com**/software/data/db2/db2connect/support.html.

Final question: How long do you plan to run your data-sharing group in coexistence? Your answer should be 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 Version 8 while minimizing any outages. After a member has been successfully migrated to Version 8, the actions of all of the members should still be the same. They should all be running the same code. So, do everything possible to keep your time in data-sharing coexistence to a minimum.

You're in Version 8 CM and don't like your access path. Now what?
Good question, and one that is of concern to almost everyone when they migrate to Version 8 (or any other version). First, check the items that IBM Support Center might ask you to check. When preparing for your Version 8 upgrade, make sure to keep the EXPLAIN (PLAN_TABLE) information from your DB2 Version 7 subsystem. After you migrate to Version 8 compatibility mode (CM), you might need that saved Version 7 PLAN_TABLE data to compare to the new EXPLAIN information you are going to produce when binding applications in CM. So, 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.

EXPLAIN is an easy and straightforward process that could be your first line of defense against any potential access-path problems. You should not use EXPLAIN by itself. You can use a tool that translates all of those values for you, such as Visual Explain. Regardless of the version of DB2 you are running on the System z platform (Version 8, Version 7, even Version 6), make sure to use DB2 for z/OS Visual Explain Version 8. You can download Visual Explain at no charge from **ibm.com**/software/data/db2/zos/osc/ve/.

The beauty of Visual Explain (VE) is its graphical representation of the access path for an SQL statement. With VE Version 8, 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 Version 8 has also improved the information that accompanies these graphical displays. With its enhanced capabilities, VE 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 Version 8 is a simple update of the Visual Explain that has been available for the last couple of years. Although it still does everything VE Version 7 did, Visual Explain Version 8 is a complete rewrite of the entire tool in Java. You might be surprised at how much more useful and intuitive VE Version 8 is compared to previous versions. For example, Version 8 allows you to display multiple query blocks at one time or a single query block, your choice. You can see context-sensitive tuning suggestions, 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. You aren't limited to using only what is displayed by the graph on the screen. You have the option in VE Version 8 to create an HTML report with just the click of a button.

VE also contains features that have little to do with explaining an access path. 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 menu. After connecting to your DB2 for z/OS subsystem, all of the following tables are created:

DSN_FUNCTION_TABLE	DSN_FILTER_TABLE
DSN_PREDICAT_TABLE	DSN_DETCOST_TABLE
DSN_STRUCT_TABLE	DSN_SORT_TABLE
DSN_PGROUP_TABLE	DSN_SORTKEY_TABLE
DSN_STATEMNT_TABLE	DSN_PGRANGE_TABLE
DSN_PTASK_TABLE	DSN_VIEWREF_TABLE

As this list indicates, Visual Explain Version 8 uses far more tables than the previous version of VE. All of this information is provided to 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 to figure out what DB2 Level 2 needs in order to work on your problem. With VE Version 8, from a menu on the tool bar, you can click 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 sent via FTP to IBM DB2 Service.

Visual Explain is free, so 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 might also want to look at APAR PQ62695 for some additional information about using the Java Universal Driver. For additional information, go to the DB2 Support Web page and search on "Visual Explain."

If you would like to take a look at a few papers on Visual Explain Version 8, see two presentations by Patrick Bossman, "DB2 for z/OS Visual Explain Version 8" and "Visual Explain for Version 8, We Found the Beef." Another presentation containing some discussion on VE is "Major Optimization Enhancements in DB2 Version 8," by Terry Purcell. You can find all of these presentations by searching the DB2 Support Web page.

In addition, the following Redbooks publications discuss the merits of Visual Explain Version 8:

- DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More, *SG24-6079, discusses the EXPLAIN improvements in DB2 Version 8 and VE.*
- DB2 UDB for z/OS Version 8 Performance Topics, *SG24-6465, has a VE discussion with a section describing how to use Stats Advisor.*

Compatibility mode: How long is long enough?

Migration to compatibility mode is complete, or almost complete. The next big question is how long you should stay in compatibility mode. It depends on a number of factors. 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 you have any problems. It depends on your "window" of availability. There are so many variables that it is difficult to recommend a specific length of time.

Unless you have hundreds of DB2 subsystems and data-sharing groups with an extremely complex QA program for moving software into production, you should stay in CM through a major event. For example, if by the completion of end-of-month (EOM) 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 (EOQ) for you, then EOQ is your major event. And so on. Make sure you also build all of those software migration blackout dates into your migration plan. If you are a retail shop 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 might not be the best time to also move DB2 to ENFM and NFM.

However you should not stay in compatibility mode for an extended length of time. It is costing you to be in compatibility mode. You have the overhead of some Version 8 function without the benefit of actually being on Version 8. One example is the SQL parser. DB2 parses SQL using Unicode UTF-8 (CCSID 1208). In CM your inputs might still be in EBCDIC. That means DB2 has to convert from EBCDIC to Unicode and then back to EBCDIC. That is costing you. After you get to NFM, 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. If you want to run and IVP, run the DB2 Version 7 IVP to validate that your migration to Version 8 compatibility mode worked as planned. And make sure your plan includes moving to ENFM and NFM, sooner rather than later.

And rest assured, this is a decision you will eventually have to make when you begin your migration to Version 8 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, there will be performance gains. Some of those gains come from the improved optimization you could realize in Version 8 if a plan or package were rebound. Most of those optimization improvements are also available in compatibility mode. 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. By now, you should be using packages. Packages were first delivered in Version 2.3. If you are using packages, your plans will be bound using the PKLIST keyword, not with DBRMs. If you have a plan with a DBRM, it might predate 2.3 and could be a candidate for rebind. You should also identify and rebind any performance-critical packages to check whether they are using the best access path. If you are still in the planning stage for your migration, 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. Having the ability to compare access paths and performance numbers between versions could save you a lot of time chasing down possible problems.

There are two reasons to avoid binding everything. 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 might not want. If it involved only one or two programs, no one would care. But a DB2 shop could have hundreds, thousands, or even tens of thousands 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 might be smarter, but might also be less forgiving. There is one potential solution, though.

IBM sells a tool, DB2 Path Checker for z/OS, that might ease the difficulty of binding all of those applications. Path Checker can scan your DBRMs and tell you which ones would get a different access path if they were bound. That means you would have to bind only the packages that would have a change in access path, rather than binding everything. For details about this tool, see the IBM DB2 Tools Web page at **ibm.com**/software/data/db2imstools/db2tools/db2pathchkr.html.

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

Overview of enabling new function mode

Enabling new function mode is the step that converts the catalog to Unicode. ENFM is also the step in the Version 8 migration process that does not easily allow for any real fallback to Version 7. Once you start the ENFM step, you should complete the process, do some testing, and move on. Days, possibly even minutes, rather than weeks or months, should be long enough. 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 Version 8 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. Stay there just long enough to make sure the conversion was successful, and that you still have a functioning catalog, then proceed to new function mode. The only reason you might 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 in ENFM because of testing.

Before you can even consider moving to ENFM, make sure that all of the members of your data-sharing group (assuming you are running in a data-sharing environment) are running in compatibility mode; there can be no more coexistence. Regardless of your environment, the catalog and directory need to be at a point of consistency before they are converted to Unicode. You can use the DB2 Quiesce utility to establish that point of consistency. To further ensure consistency, avoid making any updates to the catalog or directory while in ENFM.

If everything is in Version 8 CM and a point of consistency has been established, run an image copy of all of the catalog and directory table spaces. When the copies are complete, it's time to run the installation job DSNTIJNE. This job positions your catalog for running in full-function DB2 Version 8. Besides containing the Unicode conversion step for each catalog table space, this job stream also contains 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 data set (BSDS), make all of the necessary types and length changes to the existing catalog columns, convert the catalog data to Unicode, change the buffer pools used by certain catalog table spaces because of page-size increases, and change catalog indexes built on varying-length columns to NOT PADDED. Later sections of this paper discuss these steps.

While you are in ENFM, your database request modules (DBRMs) 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).

If you are curious about your progress in the migration process, use the DB2 command DISPLAY GROUP DETAIL to find out. In the DISPLAY output, the MODE keyword on message DSN7101I indicates your status. MODE specifies one of these values: C– compatibility mode, E– enabling new function mode, or 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; or E*, meaning you are at enabling new function mode but have been to a higher mode. This DISPLAY command also issues message DSN7133I to indicate which table spaces have been converted to Unicode.

Conversion of the catalog to Unicode

During ENFM, most of the DB2 catalog is converted to Unicode. Specifically, 17 of the 22 catalog table spaces and one directory table space are converted. Only objects in the DB2 catalog and directory are converted to Unicode. This step does not touch any user data or user-defined objects. DB2 converts only what it owns. SYSCOPY, SYSEBCDC, SYSJAUXA, and SYSJAUXB remain in

EBCDIC at the end of ENFM. Of the directory table spaces, the four that remain in EBCDIC are DBD01, SCT02, SYSLGRNX, and SYSUTILX. Keep in mind that, after the catalog has been converted to Unicode, the collating sequence is different than in EBCDIC, so some of your catalog query results might change.

As mentioned previously in this paper, the DB2 REORG utility performs the Unicode conversion. Make sure that you have run REORG against the catalog before you get to ENFM.

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

Changes to length and data type of columns in the 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 Version 8. Long names in Version 8 are going to be defined in the catalog as VARCHAR (128). You can see immediately how this is going to affect everything you do in DB2. Something as simple as a column that participates in an index key can be challenging. Now you can see the importance of the new NOT PADDED being forced to the catalog indexes by the ENFM step.

Which columns in the catalog will be longer? Mostly, they are columns that will be defined as VARCHAR (128), but there are exceptions. DB2 defines the following column types as VARCHAR (128), with the exception of the items that are denoted with an asterisk (*): 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. 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 that queries the catalog will probably 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.

Key areas to consider:

- *Although columns are defined in the catalog as VARCHAR (128), the maximum length allowed remains 30. There are still things that have to be changed in DB2 before a column will reach its full potential. (The increased column length might cause a problem for some applications.) The SQLDA column length is 30 bytes.*
- *The next exception is the package name. Every place 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, so their maximum length remains 8 bytes, although the catalog says VARCHAR (128).*
- *Locations will still have a 16-byte limit, even though the catalog defines all columns associated with a location as VARCHAR (128).*

Changes to buffer pools because of page-size increases

The change in the length of some columns during ENFM causes a change in page size to pages greater than 4 KB. With a larger page size comes the need to use a different buffer pool. As a result, job DSNTIJNE changes some of the catalog table spaces to use buffer pools other than BP0. The catalog table spaces SYSDBASE, SYSGRTNS, SYSHIST, SYSOBJ, SYSSTR, and SYSVIEWS will all have a page size of 8 KB in Version 8 and use buffer pool BP8K0. The SYSSTATS catalog table space will have an increased page size of 16 KB and will use BP16K0 in Version 8.

NOT PADDED feature and catalog indexes

The NOT PADDED feature is added to all of the catalog indexes during ENFM to avoid any problem with longer keys being built on VACHAR (128) columns. This is a "gotcha" in this process, though. If you are using user-defined indexes on the catalog, you must add shadow data sets to the DSNTIJNE job stream.

This job can convert user-defined indexes to Unicode if they have been added to the job stream. Before running the job, ensure that the shadow data sets have space for PADDED indexes. If you do not take this action before running the job, DSNTIJNE fails. After running DSNTIJNE, alter the indexes to NOT PADDED. An alternative solution is 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.

Potential trouble spots in ENFM

By taking appropriate measures before and after running the ENFM jobs, you can avoid potential trouble spots.

Everyone likes to be cautious during a migration like Version 8, so before running any of the ENFM jobs, consider running a DSN1COPY CHECK and DSN1CHKR to revalidate the catalog. Revalidating the catalog is even more important if you have been in compatibility mode (CM) for 6 months or so before moving to ENFM. By taking this precaution, you can avoid finding a bad link or corrupted page during the middle of your catalog migration.

When the ENFM job stream (DSNTIJNE) fails, the most significant reasons are lack of adequate sort space and the allocation of shadow tables using the incorrect size. Before submitting the job, check that the allocation values all seem reasonable. Pay specific attention to SPT01 and SCT02; allocating the correct sizes for these can be difficult. To ensure that you have adequate sort space, you might also want to change SORTNUM from 4 to 64 for the steps that deal with SPT01 and SCT02.

And while you are checking things out before running your ENFM jobs, ensure that DATACAPTURE is set to NO for the catalog tables.

ENFM turns off DATA CAPTURE CHANGES (by resetting it to NONE) for all catalog tables that you might have had it set on for, but does not turn it back on. That 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.

Confirm that you have addressed all of the user-defined indexes you have created on the catalog (see the earlier section on the NOT PADDED function).

When you finish running the catalog migration jobs, remember to run RUNSTATS on the DB2 catalog. The values in the STATSTIME column in the SYSCOLUMNS catalog table are invalidated during the migration. Statistics for EBCDIC values are not the same as those for Unicode UTF-8.

Changes to watch for after completing ENFM

You've just finished running DSNTIJNE and moving your DB2 subsystem or data-sharing group into enabling new function mode (ENFM). So, you run the SPUFI query SELECT * FROM SYSIBM.SYSPACKAGE and you get a package name and lots of blanks. What happened? You forgot about the conversion to long names in DB2 Version 8.

One of the tasks performed by DSNTIJNE is the conversion of certain names in the catalog to long names. Specifically, columns, mainly variable or fixed length and 18 characters long, are converted to VARCHAR columns with a maximum length of 128 characters. Actually, long names covers two different changes, each having its own concerns. First, for name-related columns, the length has increased from 18 to 128 characters. Anything displaying one of these columns needs to be aware of the longer lengths, even when doing something as simple as running a query in SPUFI. If you are not accustomed to using SUBSTR or SUBSTRING (using length units specified as CODEUNITS32, CODEUNITS16, or OCTETS) functions, then it is time to learn. You could be seeing a lot of these functions if you write your own catalog queries. You might also want to investigate the LENGTH function, in case you need to know how long one of these fields is before you use it. The second long name change is the change to VARCHAR. Not only do you have to code for length, now you have to actually extract the length so that your application can process the information correctly.

Speaking of SPUFI and longer lengths on some columns, consider making some adjustments. On the current SPUFI defaults panel, you might want to change MAX CHAR FIELD to something longer than 128; otherwise, a number of columns that now use long names might get truncated from the column's right 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 might wrap. Remember, you can scroll left and right in ISPF BROWSE.

The PLAN_TABLE is also affected by long names, because it contains columns that are also going to change to VARCHAR (128). Fortunately, you always get a sample PLAN_TABLE, so the table itself should not be a problem; however, the queries that you use against the PLAN_TABLE are affected.

As mentioned in the earlier section on ENFM, some 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 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 might need to adjust SQL that uses LIKE predicates. And there are a couple of "gotcha" 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. Author IDs are also still 8 characters, although all of the CREATOR type columns are defined as VARCHAR (128). Schema names might be in the CREATOR columns, and they can exceed 8 characters.

Most likely, when you initially upgrade to Version 8, none of this will have much of an impact, other than the SUBSTR function in SPUFI. Not many shops have many names longer the 18 characters before upgrading. But in time, you will.

Aside from long names, another change you will notice the first time you use the catalog after completing ENFM is the order of things returned. It's all going to seem backwards for a while, because your catalog is now encoded using Unicode, not EBCDIC. And with Unicode, everything is going to sort much as it does in ASCII, because the first 127 Unicode characters are the same as the first 127 ASCII characters. But there is a possible solution for that too.

For example, let's say you want to list out all of your tables but you want them returned in the same order they would have been returned back in Version 7. To return the result of this query in EBCDIC collating sequence, use these commands:

```
SELECT CAST (NAME AS VARCHAR(128) CCSID EBCDIC)
AS E_NAMEFROM SYSIBM.SYSTABLE WHERE...
```

Overview of new function mode

It's finally time to start using all of that DB2 Version 8 function you've heard so much about. The planning has been completed, you ran the IVPs and performed all of the testing under compatibility mode, and migrated the catalog during ENFM. Now you are ready to make that final move to new function mode (NFM). What can you expect and what should you do now?

After running DSNTIJNE, you are ready for your last step: running job DSNTIJNF to move to new function mode (NFM). This job calls the CATENFM utility, specifying the COMPLETE keyword to make sure DB2 is ready to move to NFM. The utility makes sure that long-name support has been added; that 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. At this point, all DB2 for z/OS Version 8 features and functions are available. However, with the ability to use all that new Version 8 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 in ENFM.

What's left to do? You can now start creating user-defined NOT PADDED indexes on the DB2 catalog, delete the VSAM data set for DSNDB06.DSNKCX01 (the index for the now-finally-departed SYSIBM SYSPROCEDURES), and run DSNJCNVB (the conversion/expansion utility) to convert the bootstrap data set (BSDS) to handle 93 active logs and 10,000 archive logs. Running DSNJCNVB requires DB2 to be stopped. After you convert the BSDS, you are ready to run the Version 8 sample jobs that are shipped with Version 8. (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. After you move to new function mode (NFM), you can use the SQL ALTER statement to reenable DATA CAPTURE on the catalog.

If you are a data-sharing shop, then at some point after entering NFM, you will also need to quiesce your data-sharing group to enable locking protocol 2, and thus improve your system's performance. 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 to 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.

Requirement: Your system must be in new function mode before you can migrate to DB2 9 for z/OS. So if you plan to migrate to DB2 9, don't stay in Version 8 ENFM.

NFM is running, now what?

After all the work you went through getting to NFM, what's the reward for your effort? Here is a high-level overview of function delivered with NFM.

Do you dabble with SQL? DB2 for z/OS Version 8 has lots of new function for you. 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, and so on) 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 can now be included in a SQL join.

There is also lots of new SQL functions in Version 8. These new functions include 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.

For a detailed look at these and other SQL and SQL-related enhancements, along with availability, usability, performance, data sharing, and distributed, see the IBM Redbooks publication *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079.

There is not a lot of documentation about which optimization enhancements are available in CM and which are available in NFM. Here are some of the things that are documented as being available only after you get to NFM, and that might affect the optimizer.

Materialized query table (MQT) is an 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 are SQL enhancements, not really optimization improvements. However, after you move to NFM, you should code the SQL for your application to take advantage of these enhancements to improve your FETCH and INSERT performance. For a discussion of these enhancements, see the related blog entries at blogs.ittoolbox.com/database/db2zos/archives/007377.asp and blogs.ittoolbox.com/database/db2zos/ archives/007383.asp.

Then there is the new locking protocol 2 for data sharing. You must be in NFM, and you have to recycle (quiesce) your entire data-sharing group to take advantage of it. 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 Version 7 and prior releases of DB2. Both of these conditions became stage 1 and indexable in Version 8. When Version 8 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 the fix for APAR PK12389, this particular enhancement has been made available in compatibility mode (CM).

If you want DB2 ODBC to support SQL statements up to 2 MB when connected to a DB2 V 8 subsystem, you must be in NFM and have applied the fix for APAR PQ88582.

If you have any DB2 ODBC applications that use an array INSERT in DB2 Version 7, the CLI/ODBC driver coverts the array INSERT into individual row INSERTs. The new multi-row syntax will be used automatically for this after you move to NFM.

After you are in NFM, you can 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.

Some more improvements you might 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.*
- Backward index scanning: *Yes, you can read an index backwards, so there is no longer a need for two separate indexes, 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 a major improvement for applications that do not support all of the data types available in DB2 for z/OS.*
- Nonuniform distribution statistics on nonindexed columns: *In Version 7, you could use the DSTATS application to turn nonindexed column statistics into columns, to enhance the optimizer's choice of access paths. That was good but it was an application running against your data. In Version 8, 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 Linux®, UNIX®, and Windows®.*
- Parallel sorting: *Version 8 attempts to take advantage of parallel sorting more often.*
- 64-bit exploitation: *Version 8 takes advantage of 64-bit memory in many ways: pools above the bar, more storage for parallel task, dictionaries, and so on.*
- Data partitioned secondary indexes (DPSI): *Secondary indexes now have full partition independence. However, using DPSI might require SQL changes to maintain the wanted access path. Also, nonpartitioned secondary indexes are now referred to as NPSI rather than NPI.*
- 4096 partitions: *By allowing one partition for every day of an 11 year period, Version 8 provides for serious size and growth for a single table.*
- 2000-byte keys: *Index keys have been increased to a 2000-byte maximum, making it easier to define large amounts of data in the index of index-only access.*
- Long object names: *Most object names in Version 8 have been increased to 128 bytes.*
- Size of SQL statements increased to 2 MB: *The 32 KB maximum SQL size in Version 7 was restrictive and prevented coding complex SQL and SQL-stored procedures. This limitation has been removed in Version 8.*

- Code-page-independent parsing: *The DB2 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 available in DB2 for Linux, UNIX, and Windows and now is available in DB2 for z/OS.*
- Recursive SQL: *Remember the old bill-of-materials type application queries that you could never do on DB2 for z/OS? Now you can, thanks to the enhancement for common table expressions.*
- Multiple DISTINCT: *Multiple DISTINCT values 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.*
- Joining up to 225 tables: *A maximum of 225 tables can now be specified in the FROM clause of a single SQL statement.*
- Materialized query tables: *This enhancement, also known as automatic summary tables or materialized views, allows the optimizer to rewrite a query to make use of a preaggregated results set.*
- Multi-row FETCH and INSERT: *This enhancement 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: *This enhancement, also known as sliding secondary extents processing, allows DB2 to manage secondary, and in some cases primary, space allocations for table spaces.*
- More log space: *The number of active logs has been increased to 93, and the number of archive logs has been 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: *Provided that you have z/OS V1.5 or later, you can now use a SYSTEM BACKUP utility and a SYSTEM RESTORE for faster recoverability of your entire subsystem or data-sharing group.*

DSNTEP4: A reason to move to DB2 Version 8 NFM

Another NFM feature you can look forward to taking advantage of almost immediately is the new sample program DSNTEP4. DSNTEP4 is an upgraded version of the well-established sample program DSNTEP2. Written in PL/I just as its predecessor, DSNTEP4 gives you multi-row fetch capabilities. 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. By using this feature, you can greatly reduce the CPU time used by DSNTEP4 and might reduce its elapsed time. Remember though, you do have to be in Version 8 NFM to use DSNTEP4 because of the multi-row fetch feature. You also need to stay current on maintenance for both DSNTEP4 and multi-row fetch. Several APARs that have been opened about DSNTEP4 and multi-row fetch fixes must be applied before you can successfully use these options.

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

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

For a description of the DSNTEP4 parameters and control cards, see the *DB2 Utility Guide and Reference*, SC18-7427.

DSNTIAUL: Another reason to move to DB2 Version 8 NFM

The DSNTIAUL sample unload program is used extensively in some accounts. When you move to NFM, you should get the updated sample for Version 8, replacing the old one. DSNTIAUL changes are similar to those in DSNTEP4. Where there are many rows to fetch, the new program, using multirow FETCH, can reduce the CPU time by about half. If you run DSNTIAUL often, you can save valuable CPU cycles.

DB2 9 benefits for DB2 Version 8 users

Surprising as it might seem, users of DB2 Version 8 (and even Version 7) can benefit from enhancements initially provided for DB2 9.

Countdown to the end of the DB2 private protocol

The talk about doing away with the DB2 private protocol has been going on forever. There have been no changes to this function since DB2 Version 4. Through all the versions since then, including Version 8, there have been subtle suggestions to start migrating your applications that use private protocol to Distributed Relational Database Architecture™ (DRDA®). Until now, your only incentive has been the lack of change for 10 years and the promise that someday private protocol will be removed.

It looks like the promise is about to be fulfilled. DB2 9 will bring you the next step toward the demise of the DB2 private protocol. If you attempt to use private protocol when binding or rebinding a plan or package (DBPROTOCOL(PRIVATE) keyword) after 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" might give you a similar function. You will be able to define a remote SNA connection as a trusted IP address and emulate "Already Verified."

For a long time, three-part name support and DRDA performance were also often considered inhibitors to moving off private protocol. However, in DB2 Version 6, three-part name support was added to DRDA so that an application could take advantage of DRDA without having to be rewritten. And the performance issues 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 non-IBM applications, there 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 do not use DRDA is fairly easy. Simply 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. APAR PK27413, with the associated PTF UK16431 (which 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, as well as 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. (Bear in mind that you need DB2 REXX Language support to use this EXEC.)

Currently, PTF UK16431 is available only for DB2 9. However, the REXX EXEC is also available to users of DB2 for OS/390 and z/OS Version 7 and DB2 for z/OS Version 8. So, there is no need to wait until you upgrade to DB2 9 to start moving off private protocol.

To download the appropriate EXEC for your particular environment, go to the IBM DB2 Support Web site and search on "Determining if plans or packages have a remote location dependency." Make sure you download the correct EXEC for your level of DB2: DSNTP2D7 is for DB2 Version 7, and DSNTP2D8 is for DB2 Version 8 compatibility mode (CM). In addition to the REXX Execs, 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

One side effect of converting to DRDA protocol is that any SQL statement that makes a reference to a remote object via an ALIAS reference in the local DB2 will not have the ALIAS reference within the statement resolved before the statement is sent on to the remote server. To address this situation, the EXEC examines the defined ALIASes within the DB2 subsystem and generates any necessary two-part name CREATE ALIAS statements, which you should then run at the location referenced by the local ALIAS.

Migration/fallback informational APAR for DB2 9

So, how is your 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)? Maybe you're just feeling adventurous. If you are, make sure to check this DB2 APAR: II12323: DB2 Version 8.1 MIGRATION/ FALLBACK INFOAPAR TO/FROM DB2 V9.1.

Yes, that is the migration and fallback APAR to get to DB2 9 for z/OS and its fix is now available, complete with a list of APARs with their associated PTFs that have already closed.

When you're checking this new APAR II12323, you might find a second DB2 9 migration and fallback APAR. Ignore it. After you are running in DB2 Version 8 NFM, get the DB2 9 checking or premigration job DSNTIJP9 (in the fix for 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.

Conclusion

There are some excellent reasons to start your Version 8 upgrade today. Here are just a few:

- *DB2 Version 7 is going out of service. On the IBM Software Support Lifecycle Web site, the end of marketing (EOM) for DB2 Version 7 was March 5, 2007 (announcement letter 906-254) and the end of service (EOS) is June 30, 2008 (announcement letter 907-023). That means EOS is about a year away. Including planning, your migration from Version 7 to Version 8 could easily take 6 to 12 months. By the way, if you are still running a version of DB2 older than Version 7, the code you are running is well out of service.*
- *IBM has delivered DB2 9. Before you can upgrade to DB 9, you have to be in DB2 Version 8 NFM and using z/OS V1.7.*
- *DB2 9 includes full pureXML™ support.*
- *Only DB2 Version 8 (and later) is enabled to use the recently announced zIIP specialty engine. A portion of DB2 Version 8 work will be eligible to take advantage of this engine. If you are still on Version 7, even if you have an IBM System z9 processor, you cannot use the zIIP engine.*

- *Here are examples of some of the z9 and z/OS features that DB2*
  - Hardware Compression
  - zIIP
  - 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
  - IPVersion 6 (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™

For more information

The IBM DB2 Web site provides all the information you need to learn about DB2 Version 8, including materials covering planning, installing, migrating, administration, testing, and other topics on the, organized by category. Go to:

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

IBM Corporation
1133 Westchester Ave.
White Plains, NY 10604

Produced in the United States of America
08-07

IMW11931-USEN-00