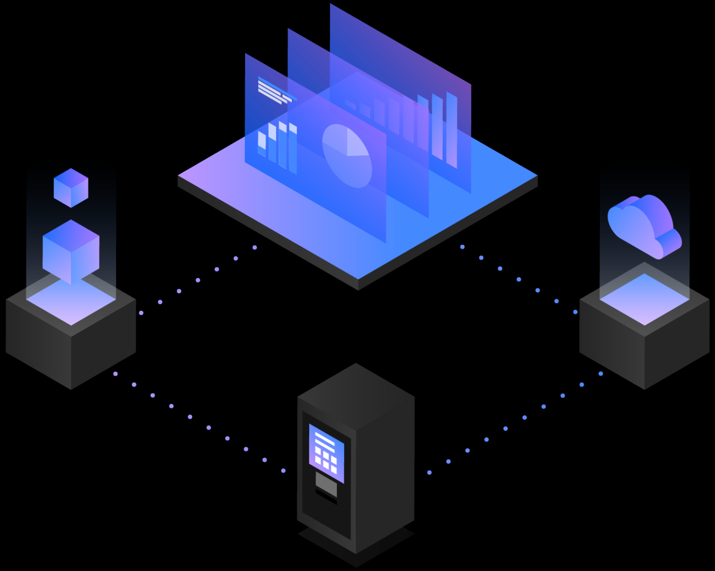
Managing Db2 for z/OS Application Compatibility Levels

TRIDEX

March 16, 2023

Robert Catterall, IBM
Principal Db2 for z/OS Technical Specialist

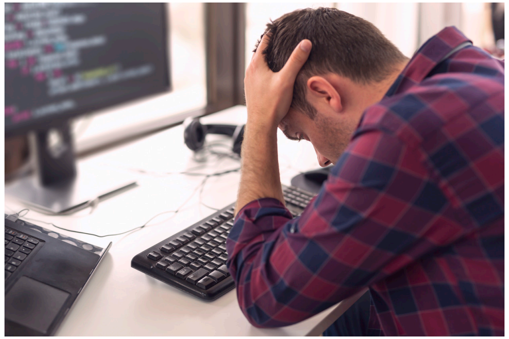


Why you want to get this right...

Facilitate innovation by development teams

Protect application quality and reliability

Avoid database administration headaches



Agenda

- What do we mean by “Db2 for z/OS application compatibility level?”
- Clearing up some misperceptions
- A special case: the IBM Data Server Driver (Db2 Connect) packages
- Avoiding database administration headaches

What do we mean by “Db2 for z/OS application compatibility level?”

It’s a package thing...

- More specifically, a Db2 package thing
- Every program that issues a Db2 for z/OS-targeting SQL statement – whether static SQL or dynamic SQL – does so by way of a Db2 package
  - For a static SQL-issuing program, you can think of its Db2 package as the compiled, executable form of the program’s SQL statements
  - For a dynamic SQL-issuing program, in most cases the Db2 package will be one of those associated with the IBM Data Server Driver / Db2 Connect
    - The IBM Data Server Driver provides JDBC and ODBC drivers (and others) that can be used by Db2-accessing programs, and SQL statements issued by way of these drivers are dynamic statements from a Db2 perspective

What is a package’s Db2 application compatibility level?

- A Db2 package executes at a certain Db2 application compatibility level
- A package’s application compatibility level is associated with a particular version and – if the version is 12 or later – function level of Db2 for z/OS
- A package’s Db2 application compatibility level has a SQL syntax consequence
  - Example: the OFFSET clause of the SELECT statement (useful for result set “pagination”) was introduced with function level 500 of Db2 12
    - If a program is going to use this clause, its package must be executing with an application compatibility level of Db2 12 function level 500 (or higher)

Not just SQL syntax...

- A package's Db2 application compatibility level also has a SQL behavior consequence
  - Sometimes, the behavior of a SQL statement changes with a new version or function level of Db2 for z/OS
  - We call such a behavioral change a "SQL incompatibility"
    - What that means: same SQL statement, same data, *different result*
- If a program would be negatively impacted by a SQL incompatibility, the problem can be negated via the application compatibility level of the program's package

As explained on the next slide…

Application compatibility level negates SQL incompatibility

- Here’s an example of a SQL incompatibility:
  - In Db2 10 system, program could cast 8-byte store clock value as timestamp:

    ```
    SELECT CAST(X’CAB5060708090100’ AS TIMESTAMP) FROM SYSIBM.SYSDUMMY1;
    ```

  - In Db2 11 (new-function mode) environment, execution of above statement would generate a -180 SQL error code, because store clock value was no longer valid input for CAST expression that references TIMESTAMP data type
- As is typical for Db2 SQL incompatibilities, this one affected few or zero programs for most organizations – *but what if you had a program that needed to cast an 8-byte store clock value as a Db2 timestamp?*
  - No problem: just make it so the program’s package will execute with an application compatibility level of Db2 10 – that means Db2 10 SQL behavior, so the CAST expression will work as it did in a Db2 10 environment

How does a package get an application compatibility level?

- Answer: from the value provided for the APPLCOMPAT option when the package was last bound or rebound
  - Example:

```
BIND PACKAGE (USIBMSTODB22.TEST) -
  MEMBER (DSN8BC12) -
  ACTION (REPLACE) -
  QUALIFIER (PRODUCTN) -
  APPLCOMPAT(V12R1M502) ⟵
```

The program using this package can use SQL syntax available with function level 502 of Db2 12, and SQL behavior for the program will be that of Db2 12 function level 502

- In a Db2 12 or 13 system, a package's APPLCOMPAT value can be V10R1, V11R1 or any Db2 12 or 13 function level – *so long as APPLCOMPAT value does not exceed Db2 system's current activated function level*

When BIND/REBIND issued <u>without</u> APPLCOMPAT value...

- In that case, will the package have an APPLCOMPAT value?
  - YES – <u>every</u> package gets an APPLCOMPT value
  - What will that APPLCOMPAT value be? It depends...
    - If BIND PACKAGE is issued without an APPLCOMPAT specification, the package's APPLCOMPAT value will be the value of the ZPARM parameter APPLCOMPAT
    - If REBIND PACKAGE is issued without an APPLCOMPAT specification, the package's <u>current APPLCOMPAT</u> value will be retained

If the package being rebound <u>does not have an APPLCOMPAT value</u> (more on that in a moment), its APPLCOMPAT value will be the value of the ZPARM parameter APPLCOMPAT

A related Db2 parameter: SQLLEVEL in module DSNHDECP

- DSNHDECP (like DSNZPARM) is a read-only module that provides values for system parameters – DSNHDECP parameters are application-related
- One of the DSNHDECP parameters, SQLLEVEL, pertains to programs that issue static SQL statements, because it relates to the <u>precompile</u> action that precedes the bind of a program's static SQL statements
- If SQLLEVEL is set to V12R1M500, and a program contains an embedded static SQL statement using syntax introduced with Db2 12 function level 501, a precompile of the program will fail
  - That being the case, you probably want the value of SQLLEVEL to be equal to the Db2 system's activated function level
  - If you bind programs with a current APPLCOMPAT value to support current SQL syntax, back-level SQLLEVEL value defeats that purpose for static SQL programs

Clearing up some misperceptions

APPLCOMPAT affects access path selection, right?

- WRONG
- Scenario: Db2 12 function level 100 introduced new access path choices for the query optimizer, including “adaptive index selection” (execution-time determination of order of index access when “index AND-ing” used)
  - To take advantage of adaptive index selection, does a program’s package have to be bound with APPLCOMPAT(V12R1M100) of higher?
  - NO – just need to bind or rebind package in Db2 12 system, *regardless of the package’s APPLCOMPAT value*
  - APPLCOMPAT relates to allowable SQL syntax and to SQL behavior – it has nothing to do with access path selection

APPLCOMPAT in <u>ZPARM</u> determines Db2 functionality, right?

- WRONG
- Scenario: Db2 12 function level 505 introduced rebind phase-in functionality (enables success of package rebind even when package is in-use at rebind time)
  - Suppose a system administrator has executed the Db2 command -ACTIVATE FUNCTION LEVEL (V12R1M505), and the value of APPLCOMPAT in ZPARM is V12R1M500 – is rebind phase-in functionality available in this system?
  - YES – APPLCOMPAT in ZPARM has **1** purpose: <u>provides default value</u> when BIND PACKAGE issued without APPLCOMPAT specification (or when REBIND issued, without APPLCOMPAT specification, for package with no APPLCOMPAT value)

A package always has an APPLCOMPAT value, right?

- NOT NECESSARILY
- Here’s the the thing: the APPLCOMPAT option of BIND and REBIND PACKAGE was introduced with Db2 11 for z/OS
  - That being the case, if a package was last bound or rebound prior to Db2 11, *it won’t have an APPLCOMPAT value*
  - You can check this in your Db2 environment: query the SYSPACKAGE table in the Db2 catalog, and see if any rows have a blank value in the APPLCOMPAT column

What if you have packages with no APPLCOMPAT value?

- My advice: if you have a no-APPLCOMPAT package and it is still used (check value in LASTUSED column for package's row in SYSPACKAGE catalog table), rebind the package to <u>give</u> it an APPLCOMPAT value
  - Why? So you'll know what the APPLCOMPAT value for the package will be if it's later rebound without an APPLCOMPAT specification: in that case, the existing APPLCOMPAT value will be retained
    - Otherwise, if a no-APPLCOMPAT package is rebound without an APPLCOMPAT specification, package will get APPLCOMPAT value from the ZPARM parameter – that could be a surprise, and I prefer a no-surprises Db2 environment
  - When you rebind a still-in-use no-APPLCOMPAT package, I'd suggest going with APPLCOMPAT(V10R1) – that would be the package's de facto application compatibility level

In Db2 13 system, no-APPLCOMPAT packages less likley

- Here’s why:
  - Db2 12 function level 510 must be activated before you can migrate to Db2 13
  - Db2 12 function level 510 will not be activated if Db2 determines that there are any packages in the system that a) have been used within the past 18 months and b) were last bound or rebound prior to Db2 11
  - A package last bound or rebound in a Db2 11 or Db2 12 or Db2 13 system will have an APPLCOMPAT value
  - So, if you have package in Db2 13 system with no APPLCOMPAT value, it must be one that was last bound or rebound prior to Db2 11 and last used > 18 months prior to activation of Db2 12 function level 510 (prior to migration to Db2 13)
    - A request to execute that package in a Db2 13 environment will result in auto-rebind (pre-Db2 11 package cannot execute in Db2 13 system), and the auto-rebind action will give the package an APPLCOMPAT value

A special case: the IBM Data Server Driver (Db2 Connect) packages

What makes these packages special?

- The IBM Data Server Driver (used by way of an IBM Db2 Connect license) provides the JDBC and ODBC (and other) drivers that Linux-, UNIX- and Windows-based applications use to send SQL statements to Db2 for z/OS
- What that means: APPLCOMPAT value for IBM Data Server Driver packages will be *default application compatibility level* for your LUW-based client-server Db2 applications (these increasingly run on cloud-based servers)
  - That determines SQL syntax available for developers of these applications
  - That determines the behavior of SQL statements issued by these applications

Data Server Driver packages: “default-default” APPLCOMPAT

- That would be APPLCOMPAT value for packages in NULLID collection, which is default collection for IBM Data Server Driver-using applications
- What should that APPLCOMPAT value be? No one right answer…
  - Some organizations like to make APPLCOMPAT for NULLID packages as current as it can be for a Db2 system (for example, if system’s activated function level is V12R1M508, APPLCOMPAT for NULLID packages will be V12R1M508)
    - Rationale: Db2 team wants developers of DRDA requester applications to have, by default, access to most current SQL functionality available on system
  - Other organizations choose a “reasonable” APPLCOMPAT value, rebind NULLID packages with that APPLCOMPAT value and stay there for a long time
    - In a Db2 12 system, APPLCOMPAT(V12R1M500) is a popular choice
    - Rationale: “If it ain’t broke…”

If APPLCOMPAT for NULLID packages not right for everyone...

- Application may need different APPLCOMPAT value vs. NULLID packages
- Recommended action in that case:
  1. BIND COPY packages from NULLID into different collection (e.g., COLL_X), and in doing that specify required APPLCOMPAT value for the packages
  2. In SYSIBM.DSN_PROFILE_TABLE, insert row for application in question (might be identified by auth ID application uses when connecting to Db2 system)
  3. In SYSIBM.PROFILE_ATTRIBUTES, insert a row, associated with profile created in step 2, that tells Db2 to issue SET CURRENT PACKAGE PATH = COLL_X when the application connects to the Db2 subsystem
- Result of action: COLL_X is default collection for application, Data Server Driver packages in COLL_X have APPLCOMPAT value application needs

  (see https://www.ibm.com/docs/en/db2-for-zos/12?topic=connections-setting-special-registers-by-using-profile-tables)

Avoiding database administration headaches

An important APPLCOMPAT change – Db2 12

- When APPLCOMPAT package bind option was introduced with Db2 11 for z/OS, it pertained only to SQL DML statements (e.g., SELECT, DELETE)
- *Starting with Db2 12,* APPLCOMPAT affects SQL DDL statements, too
- What this means: if you want to use newer CREATE or ALTER statement syntax, you need to make sure that the package through which the DDL statement is issued has the required APPLCOMPAT value
  - Often, this will be a SPUFI or a DSNTEP2 package
- Plenty of Db2 DBAs did not account for this change, and ended up being surprised by the -4743 SQL error code

```
ATTEMPT TO USE NEW FUNCTION WHEN THE APPLICATION
COMPATIBILITY SETTING IS SET FOR A PREVIOUS LEVEL
```

One example of a -4743 DBA surprise

- The statement:

```
CREATE LOB TABLESPACE XXXXXXXX
IN YYYYYYYY
USING STOGROUP ZZZZZZZ
PRIQTY 5760 SECQTY 576
ERASE NO
LOGGED
DSSIZE 4G
BUFFERPOOL BP32K
LOCKSIZE ANY
LOCKMAX 0
CLOSE NO
COMPRESS NO
```

The problem:

Specifying COMPRESS – whether YES or NO – for a LOB table space only became possible with Db2 12 for z/OS at function level 500

DBA issued statement at left through SPUFI, and the APPLCOMPAT value for the SPUFI package was <u>below</u> V12R1M500 – result was -4743 error

DBA rebound the SPUFI package with APPLCOMPAT(V12R1M500), and the statement executed without error

Some newer DDL options – get the APPLCOMPAT right!

- V12R1M500
  - ALTER/CREATE TABLESPACE with PAGENUM RELATIVE (*for relative page numbering – big benefits for universal partition-by-range table spaces*)
  - ALTER TABLE with ADD PARTITION ENDING AT(x), where x is not beyond limit of table's last partition (*insert new partition in middle of universal PBR table space*)
  - CREATE TRIGGER with a SQL PL routine in the body of the trigger (*used to create an advanced trigger*)
- V12R1M502
  - ALTER/CREATE TABLE/STOGROUP with KEY LABEL (*used to encrypt Db2 data sets using data set encryption feature of z/OS*)

And a few more…

- V12R1M507
  - CREATE OR REPLACE syntax for a stored procedure (*for more-agile deployment of stored procedures, especially native SQL procedures*)
- V12R1M508
  - ALTER TABLESPACE with MOVE TABLE *table-name* TO TABLESPACE… (*used to go from a multi-table table space to multiple universal partition-by-growth table spaces in an <u>online</u> way*)
- V13R1M500
  - ALTER TABLE with ALTER PARTITIONING TO PARTITION BY RANGE (*used to change a table from partition-by-growth to partition-by-range in an <u>online</u> way*)

A really important APPLCOMPAT-related DBA thing

- Talking about Db2 12 function level 504 and “deprecated objects”
- The scoop: When a package through which DDL statements are issued is executing at application compatibility level V12R1M504 or higher, that package cannot be used to create a traditional segmented table space
  - Package also cannot be used to create a Db2 synonym, create a hash-organized table or alter an existing table to be hash-organized
- What if the currently activated function level of your Db2 for z/OS system is V12R1M504 or higher, and you need to create a traditional segmented table space – are you stuck? You are not stuck

Creating non-universal TS when function level >= V12R1M504

- Two options:
  - Create the table space using a program (DSNTEP2, SPUFI, whatever) whose package is bound with APPLCOMPAT(V12R1M503) or lower,

    -or-

  - If using a dynamic SQL-issuing program (e.g., DSNTEP2) whose package is bound with APPLCOMPAT(V12R1M504) or higher:
    - First, issue SET CURRENT APPLICATION COMPATIBILITY = ‘V12R1M503’
    - Create the traditional segmented table space

Dynamic SQL-issuing program can change application compatibility level to value lower than – but not higher than – the APPLCOMPAT value of the package it is using

Robert Catterall

rfcatter@us.ibm.com