Extended LRSN and RBA format support in IBM Db2 for z/OS

John Campbell
Michael Dewert
Gareth Jones

IBM Db2 for z/OS Development
IBM Silicon Valley Laboratory
Executive summary

Whether you are installing a new IBM® Db2® application, designing your recovery strategy or preparing for increased workloads, your plans must account for the impact on the Db2 log. This vital feature keeps a record of every change to data within Db2 and provides the functionality needed for Db2 restart, transaction backout, and data and object recovery. Since the release of Db2 Version 1 for IBM MVS™, a relative byte address (RBA) has been used to identify the location of records within the Db2 log. The original design of the RBA, as a 6-byte value, meant that Db2 could address up to 256 GB of log data. Thirty years ago, that amount of data seemed huge, enough for the foreseeable future.

However, over recent years, it has become clear that the amount of data stored in Db2 has increased exponentially, as has the transaction volume and the number of concurrent users. At the same time, disk performance and capacity have evolved dramatically to accommodate the volume of data and to allow applications speedy and reliable access to it, with increased storage capacity and improved data transfer times. With the introduction of Db2 10 for z/OS® and the elimination of almost all virtual storage constraints, customers have taken the opportunity to vertically scale their Db2 systems even further.

These developments have made it clear that the current design of the Db2 log with a 6-byte RBA—and additionally a 6-byte log record sequence number (LRSN) in data sharing—no longer provides sufficient log capacity. The solution to this problem, as described in this white paper, is to extend the RBA and the LRSN to 10 bytes. This optional feature was delivered in Db2 11 for z/OS, giving customers the ability to choose whether to implement the extended RBA and LRSN during the lifetime of Db2 11 for z/OS.

This white paper was written primarily for Db2 for z/OS systems programmers and DBAs who are responsible for implementing the extended RBA and LRSN. It is also intended for IT executives and project managers who are involved in the decision-making and planning processes.
Appendix A. Converting a 10-byte LRSN to a time stamp .............................................. 43
Transition from the basic, 6-byte to the extended, 10-byte RBA and LRSN format

A vital requirement for every Db2 release is to ensure data integrity, data consistency and data recoverability. Therefore, Db2 records every change that is made to the data—along with other significant events—in the form of log records, which are written to the Db2 recovery log. Db2 uses relative byte addressing to uniquely identify each log record for a Db2 system or data sharing member. That is, the RBA continues to increase over the lifetime of that Db2 system. In a non-data sharing environment, Db2 uses the log RBA value to synchronize restart and recovery processing. In a data sharing environment, Db2 uses an LRSN to synchronize this processing.

The first release of Db2 for z/OS (which was then referred to as 2) defined the RBA as a 6-byte value. With the introduction of data sharing in IBM Db2 Universal Database™ for z/OS Version 4, the LRSN was introduced and also defined as a 6-byte value. For over 30 years of Db2 releases, this form of addressing has been mostly sufficient. However, with an ever-increasing application workload, a growing number of customers has exhausted the log RBA address range. At the same time, a somewhat smaller number of customers is close to reaching the end of the LRSN addressing range. The reason is because an LRSN delta was applied when Db2 data sharing was enabled to ensure that the initial LRSN value was higher than the value of the high-used RBA.

Exhausting the RBA range can be resolved, but the required actions are highly disruptive. And, no effective procedure is available for exhausting the LRSN. To address this problem, Db2 11 for z/OS introduces support for an extended, 10-byte format for the log RBA and LRSN values. By using this format, you can avoid the outages that are associated with resetting the RBA, or if you have exhausted the LRSN, with moving all the data to a new data sharing group. This extended format allows for a much greater addressing range—four billion times the log RBA range. The LRSN is now 16 million times more granular, and the LRSN range is in excess of 30 thousand years.

This white paper helps you to understand the extended format RBA and LRSN and the implications of implementing the solution or continuing with the basic format RBA and LRSN. You learn what happens when you reach the RBA limit or exhaust the LRSN. You also learn about the technical details of the extended format RBA and LRSN. You understand which objects are affected by the extended format RBA and LRSN, how to convert to the extended format, and special considerations for converting to this format.
How Db2 uses log RBAs and LRSNs

To help you understand how Db2 uses log RBA and LRSN, see Figure 1 for an overview of the Db2 processing for an SQL INSERT call.

Consider the case where a new row is inserted into a table. First, the Db2 buffer manager opens the page set, if it is not already open, and any required object-level locks are acquired. The index manager component uses the clustering index (or the first index defined on the table if no clustering index is explicitly defined) to identify a candidate data page to insert the new data row.

After the Db2 data manager detects space to insert the data row, a write intent is set to indicate that changes are being made to the data page. Also, a new unit of recovery (UR) is established, with the required transaction locks being obtained. The Db2 recovery log manager creates a log record that contains the new data row in the Db2 output log buffer. Then, the data row is inserted in the target data page, the write intent is reset to indicate that the change was completed, and the changed data page is stored in the buffer pool.

At or before commit time, the log record that is created is forced out or rather written to the current active log data set. If the updated page is in data sharing mode, and if inter-Db2 read-write interest exists in the page set, the updated page is also written to the group buffer pool (GBP). The INSERT process is now complete.
The status of all active log data sets is tracked in the bootstrap data set (BSDS) of the Db2 subsystem and in all archive log data sets (up to 10,000). At some point, the changed pages in the buffer pool are written to DASD by deferred write or by the GBP castout process. The page writes are driven by such events as reaching local or group buffer pool thresholds or the close of a page set.

For each object, the Db2 SYSLGRNX directory table tracks the range of log records for which recovery information is in the Db2 recovery log (Figure 2). That is, it tracks from the physical or pseudo open at the first update until the physical or pseudo close.

Now, let's take a detailed look at the Db2 recovery log. Each Db2 subsystem has its own BSDS, which is used by Db2 for several purposes, including the management of the log. Db2 writes log records to a set of active log data sets that it uses in rotation. Once an active log fills up, Db2 initiates an offload process to copy the data in the active log data set to an archive log data set that it creates during the offload process. After an active log data set has been offloaded, it is available for reuse. You should configure Db2 to use dual logging, where the log records are written to two active log data sets at the same time, and to offload full active logs to two archive log data sets, to ensure resilience and high availability.

The Db2 recovery log stream is spread across all active and archive log data sets. The most current log records are stored in the output log buffer, but might already be written to an active log data set. Each log record is identified by a unique log RBA value that is greater than the value of the previous log record. As the name suggests, the RBA is a count of the number of bytes from the start of the Db2 log, originating from the time that the owning Db2 subsystem or data sharing member was first started. When an active log data set fills up, a log switch occurs, and an OFFLOAD process is triggered. This way, the contents of the now full active log data set are written to a new archive log data set that is created at offload time. After the log OFFLOAD process is complete, the active log data set can be reused by the Db2 subsystem.

The concept of the RBA applies to both non-data sharing and data sharing environments. Db2 uses the LRSN to manage log records in a data sharing environment in addition to the RBA. The LRSN value is derived from a timestamp from the Sysplex Timer by issuing the store clock (STCK) instruction. This value enables Db2 to sequence log records correctly across the entire data sharing group.

The log RBA and LRSN are distinctly different. A indicates the total volume of log data that was written to the Db2 recovery log before the current log record. An indicates
approximately when a log record was written. An LRSN provides no indication of the volume of log data that was written. In data sharing, a log record has both an RBA and an LRSN, although for Db2 data sharing customers, nearly all log record handling is done by using only the LRSN.

The basic format and its limits

The total amount of log data that can be addressed by using the basic (6-byte) log RBA format is 256 terabytes (TB). Until recently, this total addressing range was sufficient for most customers. However, increases in transaction and data volumes have driven up the logging volume, meaning that an increasing number of customers has exhausted the log RBA range. In some cases, this situation can happen in a relatively short time. Current high-end Db2 customer installations can exhaust the basic log RBA range of a single Db2 subsystem within 9 - 16 months. Considering the vertical scalability improvements that were introduced with Db2 10 for z/OS, this period is expected to become even shorter in the future. With current hardware, theoretically it is possible to exhaust the log RBA addressing range of a single Db2 subsystem in less than a month.

Every time the log RBA addressing range of a Db2 subsystem is exhausted, you are required to run a well-documented procedure to reset the log RBA for that Db2 subsystem. For a non-data sharing environment, an extended application outage occurs while the reset procedure is completed, possibly seriously affecting the company’s ability to conduct its business. In a data sharing environment, business continuity can be maintained by running the reset procedure, one Db2 member at a time, while the other members continue processing.

For customers who are reaching the RBA limit, you can reset the RBA by using the procedures outlined in “Resetting the log RBA” (ibm.com/support/knowledgecenter/SSEPEK_10.0.0/com.ibm.Db2z10.doc.admin/src/tpc/Db2z_restitinglogrba.dita).

Alternatively, Db2 non-data sharing customers can move into data sharing, taking into consideration the fact that you can run your applications in the new member while you reset the originating member.

Db2 data sharing customers can add a new member, draining the workload away from the member whose log RBA is exhausted and then resetting its RBA. This approach minimizes down time and, if carefully planned, can avoid a business outage.

Both cases require careful planning and scheduling, and in any event, resetting the RBA is not always feasible in today’s continuously available environments.

Exhausting the log RBA addressing range of a Db2 subsystem is triggered by a high logging rate. Therefore, the question arises: why could a problem also exist with the LRSN for Db2 data sharing groups? As mentioned previously, the LRSN value is derived in a data-sharing environment from the Sysplex Timer. The LRSN value is the high order of the store clock (STCK) value that is obtained by Db2 from the Sysplex Timer. Because the LRSN is a 6-byte value, the last valid LRSN value represents 09/17/2042 23:53:47.370, which suggests that plenty of time remains before the LRSN is exhausted, but it is not necessarily the case.

Db2 data sharing uses the LRSN value to synchronize restart and recovery actions across all members of the data sharing group. However, when migrating a stand-alone Db2 subsystem to data sharing, the LRSN value must be greater than the highest log RBA value of the original
Db2 member. This setting ensures that the LRSN values that are generated are larger than any log RBA value that is encountered before switching to Db2 data sharing. Otherwise, recoverability will be compromised. If the highest log RBA value of the original Db2 member is greater than the current LRSN value at the time of conversion to data sharing, Db2 must add an every time an STCK value is returned by the Sysplex Timer. The LRSN delta value ensures that all LRSN values that are assigned to log records are greater than the high-written RBA of the originating member of the group. Adding this value pushes the LRSN value into the future, causing the Db2 data sharing group to reach the end of the LRSN addressing range before the year 2042.

If you defer enabling data sharing to avoid exhausting the RBA, you run the risk of swapping the problem of resetting the RBA with the more serious risk of dealing with reaching the LRSN limit. That is, the closer you are to the RBA limit, the more likely you are to run into the problem of reaching the LRSN limit long before the year 2042 if you convert to data sharing. Therefore, implement any tactical move to data sharing before the high-written RBA of the originating member causes an LRSN delta to be used that exposes you to exhausting the LRSN before you are ready to move to the 10-byte RBA and LRSN. If you are already close to the RBA limit in non-data sharing, resetting the RBA and then moving to data sharing might be better. Simply moving to data sharing does not prevent one or more individual members from reaching the RBA limit in the future.

To reset the log RBA for an individual Db2 subsystem, see “Resetting the log RBA” (ibm.com/support/knowledgecenter/SSEPEK_10.0.0/com.ibm.Db2z10.doc.admin/src/tpc/Db2z_resttinglogrba.dita).

No equivalent LRSN reset procedure is available for a Db2 data sharing group.

A performance challenge is emerging with the 6-byte LRSN. LRSN values are based on the high-order 6 bytes of the value that is returned by the STCK. That is, the LRSN is incremented only every 16 microseconds. LRSN values must be unique for any page to correctly sequence and synchronize backout and recovery operations across the data sharing group. With the introduction of faster CPU processors, Db2 can obtain duplicate LRSN values when a high update rate occurs on a single index page, data page or space map page. When Db2 detects a duplicate LRSN value, Db2 spins for up to 16 microseconds, waiting for the LRSN clock to move. This leads to increased CPU consumption, extended elapsed times, and possibly increased contention. For more information, see “LRSN spin.”

**Reaching the RBA limit**

Db2 has three thresholds in place that are important when reaching the limit of the log RBA addressing range. The first threshold occurs when Db2 first writes a log record with an RBA of ‘F00000000000’X. Db2 issues the DSNJ032I message with an alert level of WARNING, indicating that Db2 is approaching the end of the log RBA range, but processing continues without affecting the application (Figure 3).

**DSNJ032I message:** APAR PK94225 introduced logic to issue the DSNJ032I message sooner for Db2 for z/OS Version 8.1. The DSNJ032I message is issued by Db2 10, earlier releases of Db2 for z/OS, and Db2 11.
Figure 3 Threshold limits of the log RBA addressing range

The next threshold is at RBA \( '\text{FFFF00000000}' \). The Db2 subsystem has less than 32 GB of log RBA addressing range available and has reached the **soft limit**. The DSNJ032I message is issued again, still indicating an alert level of **WARNING**, but Db2 update processing is now severely constrained. The only updates that are allowed are against objects in the Db2 catalog and directory. However, the Real-Time Statistics daemon will fail when it tries to externalize the in-memory statistics. Any ROLLBACK or ABORT processing is still successful to ensure data integrity.

All other updates fail with reason code \( '00C2026D' \), which is typically associated with SQLCODE \(-904\). Application objects are accessible only in read-only (RO) mode.

At the RBA \( '\text{FFFF00000000}' \) threshold, Db2 reaches the **hard limit**, with only 4 GB of the log RBA addressing range remaining available for use by Db2. Again, Db2 issues the DSNJ032I message, but now indicates that the alert level is **CRITICAL**. All limitations imposed by the **soft limit** also apply to the **hard limit**, except that all updates, including updates to the Db2 catalog and directory, fail with SQLCODE \(-904\) and reason code \( '00C2026E' \). In addition, Db2 terminates with reason code \( '00D10251' \).

The next time the Db2 subsystem restarts, Db2 automatically executes the **RESTART LIGHT(YES)** command to resolve any outstanding URs and then shuts down again. Subsequent restarts of the Db2 subsystem are allowed only in **ACCESS(MAINT)** mode. Db2 converts any other mode that is specified on the –START Db2 command to the **RESTART LIGHT(YES)** mode. When operating in **ACCESS(MAINT)** mode, any in-doubt URs can be recovered, any objects in restricted states (such as LPL or GRECP) can be started to resolve those conditions, and any in-progress utilities can be terminated.

When you reach this point, you must reset the RBA. You can no longer update any application
data, and only the system administrator can connect to Db2. These restrictions remain in force until you reset the log RBA.

**Reaching the LRSN limit**

Db2 also has three thresholds (Figure 4) when approaching the end of the LRSN range. However, these thresholds are related to time, rather than the remaining volume of log data.

![Figure 4 Threshold limits of the log LRSN range](image)

At LRSN ‘FE35173EC1E1’X, the DSNJ034I message with an alert level of WARNING is issued for the first time. This LRSN value indicates that Db2 has one year left before it reaches the end of the available LRSN range. With an LRSN delta in place, virtual year 2042 can occur sooner than historical year 2042. As for the equivalent RBA case, processing continues as normal.

At LRSN ‘FFB20C5D6BE1’X, Db2 reaches the <2 months for the LRSN range, with less than two months remaining in the LRSN range, and issues the DSNJ034I message again with an alert level of WARNING. The restrictions are the same as the restrictions for the RBA , with user database objects restricted to read-only (RO) access.

When only two weeks are left on the LRSN range, Db2 reaches the <2 weeks at LRSN ‘FFEE65E38BE1’X. This time the DSNJ034I message with an alert level of CRITICAL is issued. And, the limitations that apply to the RBA also apply to the LRSN limit, but any attempt to start a Db2 member will fail.

After you pass the HARD LIMIT when in Db2 11 new-function mode (NFM), keep in mind several considerations for using the REORG utility on DEFINE NO objects. Db2 11 introduces the OBJECT_CREATE_FORMAT system parameter (see “Working with new objects in Db2”), which determines whether objects are allocated in the basic or extended format.
If a DEFINE NO object is allocated when the `OBJECT_CREATE_FORMAT` system parameter is BASIC, you must convert the object to the extended format by using the `REORG` utility, or the object cannot be updated. If the DEFINE NO object is allocated when the `OBJECT_CREATE_FORMAT` system parameter is EXTENDED, you do not need to run the `REORG` utility.

However, for a DEFINE NO XML table with XML versioning, before you do the allocation, use the `REORG` utility even though the object is not yet created.

The DSNJ034I message is currently not issued by Db2 10 for z/OS or earlier releases. Only Db2 11 issues this message. Therefore, if you are still running Db2 10 or an earlier release, you can use the `Print Log Map` utility, DSNJU004, to determine the highest known LRSN that is assigned to an active log (Figure 5). Then, convert that LRSN to a timestamp to estimate the amount of time left before you exceed the LRSN limit. To convert an LRSN to a timestamp, see Appendix A.

```
ACTIVE LOG COPY 1 DATA SETS
START RBA/LRSN/TIME END RBA/LRSN/TIME DATE LTIME DATA SET INFORMATION
-------------------- ------------------- -------- ----- -------------------
00002A78FFFF 2005.263 17:36 DSN=DSNC910.LOGCOPY1.DS01
00002A78FFFF 2005.263 17:36 DSN=DSNC910.LOGCOPY1.DS02
00002B167FFF 2005.263 17:36 DSN=DSNC910.LOGCOPY1.DS03
00002B167FFF 2005.263 17:36 DSN=DSNC910.LOGCOPY1.DS04
00002B167FFF 2005.263 17:36 DSN=DSNC910.LOGCOPY1.DS05
```

*Figure 5* Sample DSNJU004 (print log map) output showing the highest known LRSN for an active log

**When the hard limit is approached**

As you approach the hard limits of the RBA and LRSN ranges, you can choose from several options in Db2 11 to avoid reaching those limits:

- Non-data sharing customers have the following options:
  - You can accept a significant outage and reset the RBA, including backing up all objects, because after the RBA is reset, the objects are unrecoverable.
  - You can implement data sharing, quiesce the originating member that is approaching the RBA limit, and move all work to the new member. However, you must consider the implications if Db2 uses an LRSN delta to ensure object recoverability and understand how this change can affect you if it causes you to approach the LRSN limit.

If you defer enabling data sharing to avoid reaching the RBA limit, you risk swapping the problem of resetting the RBA with the more difficult problem of dealing with reaching the LRSN limit. The closer you get to the RBA limit, the bigger the LRSN delta will be, which ensures that the data sharing LRSN is higher than the high-used RBA. The LRSN delta is required for recoverability across the non-data sharing and data sharing boundary. The larger the LRSN delta is, the sooner you will reach the LRSN limit. The lower the RBA is when you enable data sharing, the better it is.
because it will minimize the LRSN delta if one is needed.

- For data sharing customers who are approaching the LRSN limit, you can set up a new data sharing group and copy all the data from the old group to the new group, but at the cost of a much more significant outage. This approach requires you to re-establish recoverability for all objects, accept the uncertainty of changed access path selection without a fallback option, and more. The implications of reaching the LRSN limit are severe. Few customers can tolerate the outage that is required to move the data to a new data sharing group.

- If you migrate from Db2 10 NFM to Db2 11 CM and then to Db2 11 NFM, you can convert the BSDS and page sets to use the extended, 10-byte format for the RBA and LRSN as explained in the following sections:
  - “Converting a Db2 subsystem to support the extended RBA and LRSN format”
  - “Converting the BSDS and Db2 recovery log”
  - “Converting Db2 objects to the extended format”

If you are close to the hard limit, or reach the hard limit, you might be obligated to follow the process for non-data sharing customers because sufficient log space is not available to convert to the extended format. Therefore, convert to the extended format well before you approach the soft or hard limit.

**Important:** If you have not yet migrated to Db2 11, you can choose from one of the following options to deal with reaching the RBA or LRSN limit:

- Non-data sharing customers can convert to data sharing provided that the LRSN delta is small enough to avoid exhausting the LRSN.
- Data sharing and non-data sharing customers who are about to exhaust the available RBA range can reset the RBA as described in “Resetting the log RBA” (ibm.com/support/knowledgecenter/SSEPEK_10.0.0/com.ibm.Db2z10.doc.admin/src/tpc/Db2z_resettinglogrba.dita).
- Data sharing customers who are at risk of exhausting the LRSN can copy all the data to a new data sharing group and run the applications in that data sharing group.

The next section explains how to determine the amount of space that is available and the amount of time that is left before you reach the RBA limit.

**Verifying the remaining log space and estimating the remaining time before reaching the RBA limit**

You must avoid reaching the end of the available RBA (or LRSN) range and preferably avoid the restrictions that are imposed by reaching the soft limit or hard limit. To determine the amount of log space and estimate the remaining time before you reach the end of the log RBA range:

1. Determine the remaining log capacity. Enter the Db2 **Print Log Map** utility, **DSNJU004** (Figure 6) or the **-DISPLAY LOG** command.
//PLM EXEC PGM=DSNJU004
//GROUP DD DSN=DBD1.BSDS01,DISP=SHR
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *

MEMBER *

Figure 6 Sample DSNJU004 control statement

Identify the highest written RBA value. Figure 7 shows a sample Db2 Print Log Map output, with the HIGHEST RBA WRITTEN value highlighted.

Figure 7 Sample DSNJU004 (print log map) output showing using the Db2 Print Log Map utility

The –DISPLAY LOG command output is similar (Figure 8).

Figure 8 Sample DSNJU004 (print log map) output using the –DISPLAY LOG command

2. To determine the remaining log RBA range, use the highest valid 6-byte RBA value:

\[
[\text{remaining log RBA range}] = 0000000000000000 \times [\text{HIGHEST RBA WRITTEN}]
\]

3. To calculate how much log space remains before you reach the soft limit, substitute the value for 32 GB log space left in the following equation:

\[
[\text{remaining log RBA range}] = 0000000000000000 \times [\text{HIGHEST RBA WRITTEN}]
\]

4. Identify the average number of bytes logged per day. The example in Figure 9 uses the output from the DSNJU004 utility.
Figure 9 Average number of bytes logged per day generated by using the DSNJU004 utility

From this output, you can determine the average number of archive logs that are created on a daily basis. If the number of archive logs created varies significantly from day to day, use the day with the most created archive logs for the [b] variable (number of archive logs created daily).

To calculate the number of bytes logged per day, you also need to know the number of bytes per archive log data set, which you get by subtracting the STARTRBA value from the ENDRBA value. If your archive logs vary in size, you can derive a pessimistic estimate of the amount of time that remains before you reach the soft limit. You use the maximum value of the ENDRBA value minus the STARTRBA value:

\[
\text{[average number of RBAs logged per day]} = [b] \times (\text{ENDRBA} - \text{STARTRBA})
\]

5. Calculate the remaining time before you reach the end of the RBA range:

\[
\text{[remaining time]} = \frac{\text{[remaining log RBA range]}}{\text{[average number of bytes logged per day]}}
\]

Because business workloads can vary over time, complete these steps on a regular basis to verify the remaining time before you reach the end of the log RBA range or before you reach the soft and hard limits. Establish automatic monitoring for the DSNJ032I and DSNJ034I messages and to alert you when immediate action is needed to assess how much time you have before you reach either limit or exhaust the RBA range. Also, acquaint yourself with the RBA reset procedure in “Resetting the log RBA” (ibm.com/support/knowledgelcenter/SSEPEK_10.0.0/com.ibm.Db2z10.doc.admin/src/tpc/Db2z_restitlogrba.dita), and test them regularly. If you must reset the RBA of your production systems, rehearse the process, instead of doing it under pressure when delays and errors can lead to a business outage.

LRSN spin, its effects, and how to avoid it

“The basic format and its limits” highlights the emerging challenge of LRSN spins that is caused by a combination of the basic (6-byte) LRSN format, improved processor speeds, and higher volume workloads. In this section, which explains LRSN spin in more detail, you learn how it might affect your environment.

In a non-data sharing environment, data recovery is a straightforward process because the log records that must be processed are already in RBA order.

This process is quite different in a data-sharing environment. Each member of the Db2 data sharing group has its own Db2 recovery log, consisting of the active and archive log data sets that each member controls by using its own BSDS. If Db2 needs to use log records to recover objects that were previously updated by more than one member, the RBA can no longer be used...
to guarantee that log records are applied in the correct order. Instead, Db2 uses the timestamp-based LRSN. When one member is selected to perform a recovery, it can read the BSDSs of the other members. After an image copy is restored, Db2 can allocate and read the active and archived log data sets of those other members to obtain the information that is needed to complete the recovery.

To ensure data integrity, the LRSN must guarantee that the log records for an object are sequenced correctly across the data sharing group. Because CPUs have become faster, particularly with the introduction of the IBM System z10®, the fact that the basic LRSN format is incremented only every 16 microseconds introduced the phenomenon of LRSN spin. In Db2 9 and earlier releases, Db2 had to wait or spin on the processor for up to 16 microseconds until the STCK returned a value that provided a unique LRSN. That spin not only consumed CPU time, but it also held the log latch, causing subsystems with a high logging rate to suffer from log latch contention, as indicated by latch class 19 (LC19).

Several releases of Db2 have provided relief in this area, starting with Db2 9 for z/OS. Db2 9 was the first release to introduce a feature called LRSN spin avoidance, which became effective in NFM. It enabled Db2 to use duplicate LRSN values for consecutive log updates on different pages, for example, updates to a data page and an index page. Db2 9 also removed the need to hold the log latch while spinning occurs to obtain a unique LRSN.

Db2 10 went further by supporting duplicate LRSN values for multi-row INSERT processing, even for multiple inserts on the same data page. UPDATE and DELETE processing still require a unique LRSN value. This enhancement was most effective in further reducing CPU utilization, when multi-row INSERT was operating on tables with few or no indexes.

The extended format introduced by Db2 11 effectively eliminates any possibility of LRSN spin as explained in “

| Return code 00C2026D: | An update to a table or index is not allowed because the object is in the basic format with 6-byte RBA or LRSN values. Also, the RBA or LRSN that was required for the update is near the end of the 6-byte range. Rollback and abort processing is still allowed. This reason code is typically associated with SQLCODE -904. |

Converting a Db2 subsystem to support the extended RBA and LRSN format."

The extended format for RBA and LRSN values

Db2 11 for z/OS introduces the extended (10-byte) RBA and LRSN format. This format affects several Db2 objects that require modification to handle the extended format. To convert these objects to use the format, see “

| Return code 00C2026D: | An update to a table or index is not allowed because the object is in the basic format with 6-byte RBA or LRSN values. Also, the RBA or LRSN that was required for the update is near the end of the 6-byte range. Rollback and abort processing is still allowed. This reason code is typically associated with SQLCODE -904. |

Converting a Db2 subsystem to support the extended RBA and LRSN format."
The RBA format

The RBA value is extended by four bytes on the left side, as illustrated in Figure 10.

With this change, the RBA range is now 1 Yottabyte (YB) or 1,099,511,627,776 TB. A standard commodity hard disk drive (HDD) of 2 TB capacity is typically 9.5 mm thick. A stack of these drives just under 1.25 meters in height is enough to accommodate the 256 TB of log data that is supported by the 6-byte RBA format. Consider that the distance between the Earth and the Moon is approximately 384,000 km (or 240,000 miles). Therefore, the stack of 2 TB HDDs needed to accommodate 1 YB of data would measure a staggering 13.5 times the distance between the Earth and the Moon (Figure 11).

The LRSN format

The extended LRSN format is structured differently from the extended RBA format in that it is extended on both the high-order (left) and low-order (right) sides (Figure 12).
The 1-byte addition on the high-order or left side extends the existing LRSN range to cover approximately 36,534 years, where the 3 bytes added on the right or low-order side provide a timestamp granularity of 1 picosecond. The first of these changes means the LRSN will not be exhausted for the foreseeable future and beyond, and the second change eliminates LRSN spin.

The LRSN can be extended this way by using the high-order 10 bytes of the 16-byte value returned by the store clock extended (STCKE) instruction, instead of the standard STCK instruction.

**Db2 objects affected by the extended format**

The implementation of the extended format is not apparent from the perspective of the application and of normal Db2 operation. In particular, applications cannot detect whether Db2 is using basic or extended format log records.

Technically you do not need to implement the extended RBA and LRSN format in Db2 11 NFM unless you are approaching the end of the basic LRSN or need to avoid resetting the RBA. Therefore, you can decide when to convert to the extended format, if at all. However, some Db2 11 objects must be updated to be ready for the conversion process if and when it occurs. This update happens during enabling-new-function mode (ENFM) or NFM migration processing.

This section provides a brief overview of the affected Db2 objects, including those objects that need preconditioning to support extended format log records.

The RBA and LRSN values are stored in the following Db2 objects:

- The shared communication area (SCA)
- The Db2 catalog and directory
- The BSDS
- The Db2 recovery log
- Db2 application objects

**Shared communication area**

In a Db2 for z/OS data-sharing environment, the coupling facility houses three Db2 structures: the lock structure, the group buffer pools, and a list structure, called the SCA. The SCA is the only one of these structures that must be modified during the ENFM or NFM migration process to prepare for the extended format LRSN and RBA.

The SCA is used to coordinate data recovery across a Db2 group. Therefore, it contains
information about Db2 resources and objects within a data sharing group together with their status, including the following information:

- Member names
- BSDS names
- Database exception and status conditions for objects and members in the group
- Recovery information, such as log data set names and the list of in-doubt XA transactions

This information always includes LRSN and RBA values. A large number of such values can exist, depending on the number of objects or resources in exception states. The information in the SCA is critical for recovery and startup processing.

During NFM activation of a data sharing group, the **DSNTIJNF** job automatically converts the SCA to support the extended RBA and LRSN format by rebuilding the SCA. A group restart in any mode later than Db2 11 conversion mode (CM) also triggers the rebuild of the SCA to support the extended format if it is still in the basic format.

The rebuild of the SCA occurs when migrating from Db2 11 CM to Db2 11 NFM. For the user, it is not apparent and does not need any manual intervention.

**Db2 catalog and directory objects**

Conversion of the Db2 catalog objects to support the extended format is a two-phase process.

The metadata that describes the catalog itself and all user objects is stored in rows and columns in Db2 catalog tables. Some catalog tables contain information about RBAs or LRSNs and must be altered so that they can support the basic format and the extended format. For documentary and functional reasons, the catalog is also modified to indicate which format an object is in.

The following catalog and directory objects, among others, contain the RBA and LRSN information:

- SYSIBM.SYSCHECKS
- SYSIBM.SYSCOPY
- SYSIBM.SYSINDEXES
- SYSIBM.SYSINDEXPART
- SYSIBM.SYSINDEXSPACESTATS
- SYSIBM.SYSTABLEPART
- SYSIBM.SYSTABLES
- SYSIBM.SYSTABLESPACESTATS
- SYSIBM.SYSOBDS
- SYSIBM.SYSLGRNX
- SYSIBM.SYSUTILX

The **DSNTIJEN** job changes the definitions of the affected columns, which is done one catalog table at a time. If the **DSNTIJEN** job is paused or cancelled before completion, the definitions of the column of some Db2 catalog objects are already altered, while others remain untouched.

The RBA and LRSN conversion of the Db2 catalog objects during ENFM affects only the columns that store information about RBAs or LRSNs. The physical structure of the page sets that the catalog describes is not converted, and the page sets remain in the basic format at the
end of ENFM processing.

To convert the Db2 catalog and directory table spaces and index spaces to the extended format, you can run the DSNTIJCV job any time after you activate Db2 11 NFM. However, careful planning is required because the DSNTIJCV job runs the online REORG utility against the catalog table spaces. The REORG options enable use of the extended RBA or LRSN by restructuring the control information in the target table space page headers, page trailers and so on.

**Bootstrap data set**

The BSDS contains an inventory of all the active and archive log data sets that are known to the Db2 subsystem. Db2 uses this information to locate log records to satisfy log read requests during normal Db2 system activity and during restart and recovery processing. When in Db2 11 NFM, optionally, you can convert the BSDS to support the extended format by running the DSNJCNVT job.

In a non-data sharing environment, BSDS conversion requires an outage. In a data-sharing environment, you can adopt a member-by-member approach, without affecting application availability during the conversion process. Db2 continues to operate as normal, regardless of the mix of extended format and basic format BSDSs within the data sharing group.

**Db2 recovery log**

No special conversion process is needed to convert the Db2 recovery log to support the extended format. After the BSDS of a Db2 subsystem or member is converted to the extended format by running the DSNJCNVT utility, that Db2 subsystem or member will generate log records in the extended format. Log records that were previously written to the active log or offloaded to the archive log are not converted. Db2 can work with both formats. For example, a single RECOVER utility will successfully process log records in both formats.

Because an extended format log record is longer than a basic format log record, you will fill the active log data sets more quickly and produce more archive logs. Therefore, you should plan for increased disk and possibly tape storage requirements. If you typically write short log records, the increase in the logging volume can be significant. Observed increases in the logging volume are in the range of 3 percent - 40 percent.

**Db2 database objects**

The Db2 application objects can also be converted to take advantage of the extended format. Compared to the objects previously discussed, it potentially takes a much longer time to convert the Db2 database objects because of the number of objects in a Db2 subsystem. Where system-related objects can be converted at the end of ENFM or shortly afterwards, it might take months or years before all Db2 application objects can be converted.

Implementation of the extended format RBAs and LRSNs is not apparent from the perspective of the application and normal Db2 operations. Db2 11 functions with a mixture of objects in the basic and extended formats.
How Db2 11 uses the extended RBA and LRSN format

Db2 ensures data integrity when operating with any combination of the RBA and LRSN formats.

Db2 11 is fundamentally an extended, 10-byte RBA and LRSN system. All control blocks and values are processed internally in the extended format. One reason for this method is to avoid the complexity of having dual code paths to deal with both the basic and extended formats. Therefore, in all Db2 11 modes, RBA and LRSN values are displayed in the extended format. This format is independent of the migration status of the catalog and directory or whether the database page sets or the BSDS have been converted to the extended format.

For recovery purposes, the extended format is the recommended input format for Db2. The reason is not technical because the RECOVER utility supports both formats. Rather, consistently using the extended format simplifies such processes as recovery automation and copy and paste operations. When extended format RBA or LRSN values are specified as input to the Db2 RECOVER utility, conversion to the basic format is performed internally by Db2 if the object being recovered is still in basic format.

Db2 11 always works internally with extended LRSN values, which are generated by the Db2 Recovery Log Manager as shown in Figure 13.

![Figure 11 Extended LRSN values generated by the Db2 Recovery Log Manager](image)

If the log record is for a basic-format object, Db2 converts the 10-byte value into a valid 6-byte LRSN by truncating the existing 10-byte value on the left by one byte and on the right by three bytes (Figure 14).

![Figure 12 Truncated 10-byte value LRSN to a 6-byte value](image)

When converting a 10-byte value to a 6-byte value, if the LRSN value is outside the valid 6-byte range, an exception condition is generated. It imposes the following limitations, which are the same as the limitations you would encounter if you reached the hard limit in a previous release of Db2:

- UPDATE is not possible.
- ROLLBACK and ABORT processing are still allowed.
- Return code 00C2026D is issued.
- The Db2 object is placed in read-only mode until it is converted to the extended format.

**Return code 00C2026D**: An update to a table or index is not allowed because the object is in the basic format with 6-byte RBA or LRSN values. Also, the RBA or LRSN that was required for the update is near the end of the 6-byte range. Rollback and abort processing is still allowed. This reason code is typically associated with SQLCODE -904.
Converting a Db2 subsystem to support the extended RBA and LRSN format

Before you enable the extended RBA and LRSN format, you must ensure that your Db2 11 maintenance is current. The APAR keyword, EXTENDEDRA, was created to enable you to identify maintenance that is related to the extended RBA or LRSN.

Converting a Db2 subsystem to operate with the extended log RBA and LRSN format is a straightforward process. As Figure 15 shows, the conversion process depends on the migration to Db2 11 NFM. In NFM, you can choose the order in which to convert the BSDS and the database objects.

![Figure 135 Converting a Db2 subsystem to operate with the extended log RBA or LRSN format requires migration to Db2 11 NFM](image)

Converting the BSDS and the database objects to use the extended log format is optional. You do not need to start this task before you migrate your environment to the release after Db2 11. If you are not facing the prospect of exhausting the log RBA or LRSN range, you can choose to stay in the basic log format if you are in Db2 11.

Even if you decide not to move to the extended log format, you must prepare Db2 for the extended format as part of the process to migrate Db2 10 NFM to Db2 11 NFM. In Db2 11 NFM, you can decide whether and when to convert the BSDS and database objects to use the extended log format. However, although you can decide on the order of conversion, you must use the following migration strategy:

1. Run the premigration jobs and cleanup steps (see 2 11 / , GC19-4056, and “Installing and migrating Db2”  
2. Migrate Db2 10 CM to Db2 11 CM.
3. Migrate Db2 10 NFM to Db2 11 NFM.
4. Convert all BSDSs of the data sharing group within a specific number of weekends.
5. Reorganize all Db2 Directory and Catalog page sets to the extended RBA or LRSN format.
6. Set the `OBJECT_CREATE ZPARM` and `UTILITY_CONVERSION ZPARM` to `EXTENDED`.
7. Reorganize all objects to the extended RBA or LRSN format. Monitor the progress of the table space conversion process by querying the Db2 catalog regularly.
8. Set the `OBJECT_CONVERSION ZPARM` to `NOBASIC`. 
Several changes are automatically made during the migration from Db2 10 NFM to Db2 11 NFM. The biggest change occurs as soon as you start Db2 11 CM. All control blocks and internal structures are processed internally in the extended log format, meaning that Db2 messages always show RBA and LRSN values in the extended format. Db2 messages report 6-byte LRSN values as 10-byte values with a precision of 000000. The only exception is the QUIESCE utility, which externalizes LRSN values in the extended format with a precision of nnnnnn. RBA values have a leading 4 bytes of zeroes.

This change ensures that, regardless of the Db2 mode, your automation and message-handling routines do not have to factor in whether the log records written for a database object are in the basic or extended format or whether the BSDS has been converted. This change also affects the Db2 RECOVER utility, which accepts an extended format RBA or LRSN as input. If you supply a 6-byte value (for example, for TORBA or TOLOGPOINT), the utility automatically converts it to the 10-byte format, which is always used internally by the RECOVER utility.

You must plan for this conversion before you migrate Db2 10 CM to Db2 11 CM. All third-party tools and maintenance procedures must support the Db2 11 extended format. If you use the instrumentation facility interface (IFI) to read log records, you must change your routines to handle the log record format that is returned by the IFI:

- Reading complete log data (IFCID 0306)
  Db2 always returns new format log records regardless of the format that is being used by the physical Db2 log.

- Reading specific log records (IFCID 0129)
  The log capture exit and the stand-alone log read routines always detect the format of the log records.

In Db2 11 CM, the SYSIBM.SYSTABLEPART and SYSIBM.SYSINDEXPART tables have the RBA_FORMAT column, which contains one of the values shown in Table 1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Basic RBA or LRSN format</td>
<td>Undefined, DEFINE NO was specified, which is not an XML table space with XML versions</td>
</tr>
<tr>
<td>E</td>
<td>Extended RBA or LRSN format</td>
<td>For migrated objects</td>
</tr>
<tr>
<td>U</td>
<td>Undefined, DEFINE NO was specified, which is not an XML table space with XML versions</td>
<td></td>
</tr>
</tbody>
</table>

The value is set to B for newly defined objects in Db2 11 CM, or for objects that are reorganized or loaded with the REPLACE option.

**Db2 11 coexistence with Db2 10**

Running Db2 11 CM in coexistence mode with Db2 10 NFM with respect to the extended log format is fully supported. However, Db2 11 CM members can issue messages in the extended format, where Db2 10 NFM members can still issue messages in the basic format.

Any conversion that is required to generate the extended format is done internally by Db2 11 CM. However, the original values of the Db2 catalog, recovery log, or Db2 user objects are still 26
saved in the basic format.

Two APARs, PM90175 and PM90247, were introduced to support coexistence between Db2 10 NFM and Db2 11 CM. With the fixes for these APARs applied, Db2 10 running in coexistence mode with Db2 11 CM ensures that the LRSN values that are provided as input to the `RECOVER` utility are valid.

The sanity checks that are introduced with these APARs prevent Db2 from using invalid LRSN values for point-in-time recovery actions, that is, 6-byte LRSN values with a leading byte of zeros. Therefore, they prevent the `RECOVER` utility from failing with return code 8 in the UTILINIT phase (Figure 16).

```sql
RECOVER TABLESPACE DSNDB06.SYSTSTPT TOLOGPOINT X'00CB378D5A99'
DSNU510I =D2T1 105 20:27:59.87 DSNUCASA - NO RECOVERY BASE AVAILABLE FOR RECOVERY OF TABLESPACE DSNDB06.SYSTSTPT
DSNU017I  105 20:27:59.87 DSNUGBAC - UTILITY DATA BASE SERVICES MEMORY EXECUTIONABENDED,
REASON=X'00E40119' CAUSE=X'E4D9D4D3'
RECOVER TABLESPACE DSNDB06.SYSTSTPT TOLOGPOINT X'00CB378D5A9900000000'
```

Figure 16 RECOVER TABLE SPACE utility output when failing in the UTILINIT phase

**Migrating from conversion mode to enabling-new-function mode**

The changes that are required to enable the Db2 catalog and directory to support the extended log format occur during the ENFM process. The Db2 catalog and directory columns that contain the RBA or LRSN values are changed from 6 bytes to 10 bytes in length as part of the `DSNTIJEN` job.

ENFM is not an atomic process, but can be interrupted. Therefore, you might have a mix of 6-byte and 10-byte columns in the catalog and directory, depending on which job steps were successfully completed by the `DSNTIJEN` job. Db2 11 handles this mix of columns seamlessly.

**DSNTIJEN job:** For more information about the `DSNTIJEN` job, see “Job DSNTIJEN” (ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.Db2z11.doc.inst/src/tpc/Db2j obdsntijen.dita?lang=en).

Superficially, the changes are trivial, but you must be aware of some important points. By looking at how the conversion process affects SYSIBM.SYSCOPY, you can understand in more detail what happens when a catalog column is altered. For SYSIBM.SYSCOPY, the `START_RBA` column of the affected row contains the RBA or LRSN of a point in the Db2 recovery log.

**Types of Db2 recovery log points:** The type of point depends on the ICTYPE value that is also stored in SYSIBM.SYSCOPY. For more information about the `START_RBA` value, see “SYSIBM.SYSCOPY table” (ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.Db2z11.doc.sqlref/src/tpc/Db2z_sysibmsyscopytable.dita?cp=SSEPEK_11.0.0%2F10-0-30t).

Before the `CATENFM` (DSNTIJEN) utility successfully completes its update of SYSCOPY, a SELECT, which includes `START_RBA`, returns a 6-byte value. After completion, it returns a 10-byte value. However, in all Db2 11 migration modes, the `START_RBA` value is always reported
in utility output as a 10-byte value.

After conversion, LRSN values have a leading byte of 00 and a precision of 000000. RBA values are right justified with four leading bytes of 00000000, which is true for all affected catalog and directory columns. However, any new records that are written to SYSCOPY after the column was altered are stored with a precision of 0000000000 for LRSN values. The precision is unrelated to the conversion of the BSDS or the conversion of individual database objects to the extended format.

After conversion, the SYSIBM.SYSLGRNX RBA and LRSN values are stored in the extended format, even though the BSDS is not yet converted and Db2 is still logging in the basic format.

Running Db2 11 NFM with support for extended RBA and LRSN values

To run Db2 11 NFM where support for 10-byte RBA and LRSN values is enabled, you need to run the DSNTIJNF job. After the DSNTIJNF job has completed, Db2 runs in NFM mode. In Db2 data sharing, the first time you run the DSNTIJNF job, it rebuilds the SCA after converting the SCA structure to contain the 10-byte RBA and LRSN values (Figure 17). If duplexing is enabled for the SCA structure, you must disable it before running the DSNTIJNF job. Otherwise, the job fails. After the job has completed, you can re-enable duplexing.

Db2 is now enabled to work with the extended log format. However, you must still convert the following object types to fully use the extended format:

- Bootstrap data set
- Implicitly, the Db2 recovery log
- Db2 database objects

The order in which you convert these objects is up to you. Some considerations are associated with either approach and are described in the following section. The approach to conversion that is described, including the order in which the objects are converted, is based on the best
practices of customers who participated in the Db2 11 Early Support Program. Before finalizing your strategy, you should read the section on Conversion strategies which outlines recommended best practice for three scenarios: non-data sharing, to handle approaching the end of the log RBA range; data sharing, to handle approaching the end of the LRSN range; and data sharing, to handle approaching the end of the log RBA range on one or more members.

Working with new objects in Db2

The Db2 11 `OBJECT_CREATE_FORMAT` subsystem parameter controls the format that a new object is created in. Two settings are possible, `BASIC` and `EXTENDED`, that apply equally to table spaces and index spaces.

When the `OBJECT_CREATE_FORMAT` subsystem parameter is set to `BASIC`, new objects are in the basic format. When this subsystem parameter is set to `EXTENDED`, which is also the default setting, new objects are in the extended format. The new `SYSIBM.SYSxxxPART.RBA_FORMAT` column is set to E for a new object that is created in the extended format and B when the object is created in the basic format.

If this subsystem parameter setting is set to a value other than `BASIC`, it is ignored in Db2 conversion mode. However, if it is set to `EXTENDED` during ENFM processing, all newly created objects are in the extended format.

An exception is WORKFILE objects, which are always created in the extended format if they are allocated in Db2 11 CM. WORKFILE objects are not logged objects, recovery LRSNs and RBAs are not recorded in the Db2 log for work files, and RBA and LRSN data is not maintained in workfile header pages or page headers.

If you are running Db2 10 and Db2 11 in coexistence, with Db2 10 NFM and Db2 11 CM members in the same group, Db2 10 and Db2 11 members have no sysplex query parallelism support if work files are involved. Db2 11 no longer supports sysplex query parallelism, which you should avoid when running Db2 in coexistence.

Converting the BSDS and Db2 recovery log

When in NFM, the Db2 catalog and directory indicate whether database objects are using the extended log format or are continuing to use the basic format. For data sharing customers, the SCA has also been converted to work with the extended format.

However, even in Db2 11 NFM, Db2 still creates log records in the basic format. Not until the BSDS is converted does Db2 create log records in the extended format. If you decide not to convert the BSDS during the Db2 11 lifecycle, the Db2 recovery log continues to use the basic format at all times.

**Important:** Converting the BSDS is a one-way process. No utility is available to convert the BSDS back to the basic format.

Conversion of the BSDS is performed by the stand-alone `DSNJCNVT` utility, which requires that Db2 is not running during utility execution. The `DSNJCNVT` utility can run only when in Db2 11 NFM. Otherwise, it issues the DSNJ439I message and terminates without updating the BSDS.
The **DSNJCNVT** utility requires at least two data sets for the conversion. The first data set, SYSUT1, specifies the old BSDS that is used as input for the conversion. The second data set, SYSUT3, specifies the new, converted BSDS. The BSDS cannot be converted “in place.” Therefore, the output BSDS must not be the same data set as the input BSDS.

If you duplexed your BSDSs, you can use SYSUT2 to specify the second old copy of the BSDS. The converted, new second copy is specified on the SYSUT4 DD card. The SYSUT2 and SYSUT4 DD statements are optional.

The RACF user ID that is used to run the **DSNJCNVT** utility must have read and write access to the new BSDS data sets, in addition to read access to the old BSDS data sets.

In a data sharing environment, run the **DSNJCNVT** utility on one member at a time, with each member having its own BSDS. Provided that no single-member application affinities exist, this method ensures continuous application availability as you roll the change across the group, one member at a time, stopping each member as you convert the BSDS.

Keep in mind the following considerations when running the **DSNJCNVT** utility:

- Some Db2 utilities, such as the **RECOVER** and **REORG** utilities, access the recovery log of peer members in READ mode. Therefore, you need to make sure that no active **RECOVER** or **REORG** jobs are running when the BSDS conversion is running.
- Stop any data replication processes that access the log of the Db2 member that owns the BSDS that is being converted. This approach ensures that you are able to rename the BSDS VSAM data sets and run the conversion job. Then, stop the Db2 subsystem, allowing peer data sharing members to deallocate the BSDSs when the state of the owning Db2 member changes to “inactive.”

Db2 11 provides the **DSNTIJCB** sample migration job in the SDSNSAMP library for BSDS conversion. **DSNTIJCB** consists of the following job steps:

**DSNTDEF**

Uses IDCAMS to define new BSDS data sets with CLUSTER, DATA, and INDEX components under the temporary names `prefix.BDS01.NEW` and `prefix.BDS02.NEW`.

**DSNTCNVT**

Runs the **DSNJCNVT** utility to read records from the existing BSDS, converts them to support the extended RBA and LRSN format, and writes them to the `prefix.BDS01.NEW` and `prefix.BDS02.NEW` data sets.

**DSNTRENO**

Renames the existing BSDS data sets to `prefix.BDS01.OLDFMT` and `prefix.BDS02.OLDFMT`.

**DSNTRENN**

Renames the `prefix.BDS01.NEW` data set to `prefix.BDS01`, and renames the `prefix.BDS02.NEW` data set to `prefix.BDS02`.

The Db2 recovery log does not require a special conversion task. After the BSDS is successfully converted, the owning Db2 subsystem immediately starts creating log records in the extended format. Previously created log records are not converted and remain in the basic log format. Db2 supports both formats and will convert them internally if required.
LRSN spin can still occur after the BSDS is converted. To achieve LRSN spin avoidance, a true 10-byte system is required. That is, the Db2 recovery log—and therefore the BSDS—must be in the extended format, as must the Db2 database object that is causing the LRSN spin.

Figure 18 illustrates the situation after the BSDS is converted. Db2 now creates extended format log records and stores them in the 10-byte format in the Db2 catalog. RBA and LRSN values are only truncated when they are externalized in Db2 user objects.

Figure 15 Representation of LRSN values in New Function Mode after BSDS conversion but without database object conversion

After the BSDS is converted to the extended format and Db2 is restarted, you cannot fall back to a basic format BSDS. You must first restore the old copy of the BSDS that is used as input to the DSNJCNV utility, which is effectively a point-in-time copy of the BSDS. Do not restore an old copy of the BSDS. The restore entails a complex series of actions to synchronize the status of the BSDS with all the other Db2 objects, including the recovery log, the catalog and directory, and the user data. The restore process does not affect any other form of standard Db2 recovery. Nor does it affect your ability to convert an extended format table space back to a basic format table space.

Although this white paper does not address dormant data sharing members, remember to keep the BSDS and active log data sets of any dormant members. Start these members whenever you cross a release boundary, taking any necessary steps to upgrade the BSDS. Also, start any dormant members when you implement the extended RBA or LRSN format, and upgrade each BSDS by using the DSNJCNV utility to ensure that it also supports the extended RBA or LRSN format.

Considerations when installing a new Db2 subsystem
When installing a new Db2 subsystem, the BSDS is initialized in the basic format by the DSNTIJID installation job. Therefore, even for a new Db2 installation, you need to run the DSNTIJCB job to convert the BSDS to the extended format.

This process provides a consistent experience for Db2 11 NFM customers, whether they have built a new Db2 11 installation or have migrated from Db2 10. It also provides the ability to use
the **DSN1COPY** utility to copy data from a Db2 10 or earlier subsystem into any new or migrated Db2 11 system.

**Converting Db2 objects to the extended format**

The last task that remains before you completely convert your Db2 subsystem to fully use the extended format is to convert the Db2 user database objects. However, this task might take the longest time to complete.

---

**Important:** You can convert the Db2 catalog or directory and user objects from the basic format to the extended format and vice versa.

---

The amount of time that is available to convert the database objects is unlimited. You can convert all the objects over a weekend, with a batch stream scheduled to run over a couple of days, or on an on-going basis as part of your regular scheduled database housekeeping processes.

To convert the Db2 table space and index space objects, you can run the following utilities with the new **RBALRSN_CONVERSION** keyword:

- **REORG TABLESPACE** or **REORG INDEX**
- **LOAD REPLACE**
- **REBUILD INDEX**

The new **RBALRSN_CONVERSION** keyword specifies the target format for the object. It accepts the following parameters:

- **NONE**
  - Specifies that no conversion be performed.
- **BASIC**
  - Converts any object that is in the extended format to the basic format.
- **EXTENDED**
  - Triggers conversion to the extended format when an object is in the basic format.

After a Db2 page set is converted to the extended format, the **RBA_FORMAT** column in the corresponding SYSIBM.SYSxxxPART catalog table entry contains the value `E`. From this point on, the extended RBA and LRSN values are written to the page set. In a data sharing group, the precision of an LRSN value written to PGLOGRBA in the page header and header page depends on the format of the BSDS of the Db2 member that is driving the update.

LRSN with precision `nnnnnn` is written if the updating member owns a BSDS that has been converted to the extended format. Otherwise, an LRSN with precision `000000` is written if the updating member owns a basic format BSDS.

Where the new **RBALRSN_CONVERSION** keyword provides granular control for conversion at an object level, the **LISTDEF** utility has also been enhanced to support the conversion process. You can specify which objects should be included in the list on the basis of whether they are in the basic or extended format.

**BASIC** and **EXTENDED** are two new **LISTDEF** keywords. They both accept a parameter value
This filtering takes place last, after the LISTDEF utility has processed all other INCLUDE and EXCLUDE clauses.

Figure 19 shows a sample LISTDEF control statement, which causes the specified table space objects to pass LISTDEF filtering if they are still in the basic format. Otherwise, REORG processing is not triggered.

```
LISTDEF REORG_LIST
  INCLUDE TABLESPACE DSNDB01.DBD01 BASIC YES
  INCLUDE TABLESPACE DSNDB01.SPT01 BASIC YES
  INCLUDE TABLESPACE DSNDB01.SCT02 BASIC YES
REORG TABLESPACE LIST REORG_LIST SHRLEVEL CHANGE
  LOG NO COPYDDN(SYSCOPY)
  RBALRSN_CONVERSION EXTENDED
  RETRY 255 TIMEOUT TERM RETRY_DELAY 1 DRAIN_WAIT 1
  SORTDATA
```

![Figure 16 Sample LISTDEF and REORG control statement](image)

**Controlling conversion on a subsystem level**

The RBALRSN_CONVERSION utility keyword is perfectly suited for an object-level approach to conversion. Combining this capability with the filtering enhancement to the LISTDEF utility means that you can easily convert medium-sized lists of objects. However, this approach might not be the best one if you want to convert all Db2 user objects without modifying your utility control cards.

The solution is to use the Db2 UTILITY_OBJECT_CONVERSION subsystem parameter. The setting of this parameter controls the basis on which Db2 utilities that accept the RBALRSN_CONVERSION keyword perform format conversion for existing table spaces or indexes. Table 2 shows the acceptable values for the UTILITY_OBJECT_CONVERSION subsystem parameter.

**Table 2 Acceptable values for the UTILITY_OBJECT_CONVERSION subsystem parameter**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC</td>
<td>Automatically converts existing objects to the basic format.</td>
</tr>
<tr>
<td>EXTENDED</td>
<td>Automatically converts existing objects to the extended format. Is only allowed when OBJECT_CREATE_FORMAT = EXTENDED.</td>
</tr>
<tr>
<td>NOBASIC</td>
<td>Disables the RBALRSN_CONVERSION = BASIC setting. Is only allowed when using the OBJECT_CREATE_FORMAT = EXTENDED setting.</td>
</tr>
<tr>
<td>NONE (default)</td>
<td>Does not perform automatic conversion.</td>
</tr>
</tbody>
</table>

By appropriately setting the UTILITY_OBJECT_CONVERSION subsystem parameter, you eliminate the need to rewrite your utility jobs to perform object format conversion.

In most cases, the RBALRSN_CONVERSION keyword takes priority over the UTILITY_OBJECT_CONVERSION subsystem parameter if it is specified in a utility control statement. A special case is the NOBASIC setting. This setting causes the utility to fail if the RBALRSN_CONVERSION keyword is set to BASIC or if it is set to NONE and the objects are still in the basic format.
Just as the **RBALRSN_CONVERSION** keyword is only valid in NFM, the **UTILITY_OBJECT_CONVERSION** subsystem parameter is ignored in Db2 11 CM. However, if this subsystem parameter is set to **EXTENDED** when you run ENFM, it affects **CATENFM REORG** processing.

Db2 catalog column definitions are changed to support the extended format during ENFM. When you set the **UTILITY_OBJECT_CONVERSION** subsystem parameter to the default value of **NONE** or to **BASIC**, ENFM does not change the format of the underlying catalog table and index space objects. However, if the **UTILITY_OBJECT_CONVERSION** subsystem parameter is set to **EXTENDED**, all the catalog and directory objects that were reorganized during ENFM are converted to the extended format in addition to the other changes made by ENFM.

If you have a limited change window for migration, keep this limitation in mind, because format conversion for the catalog and directory table and index spaces might cause elongated elapsed utility times.

Figure 20 shows how LRSN values are represented for Db2 user page sets that have been converted to the extended format. In this example, because the BSDS has not been converted, the LRSN is truncated to 6 bytes when it is written to the Db2 recovery log.

**Conversion strategies**

As well as considering all the technical details discussed above when devising a strategy for converting to a 10-byte log RBA or LRSN system, you also need to take into account the particular conversion scenario: non-data sharing, to handle approaching the end of the log RBA range; data sharing, to handle approaching the end of the LRSN range; or data sharing, to handle approaching the end of the log RBA range on one or more members, when you are not also approaching the end of the LRSN range.
This section describes a safe conversion strategy for each of the three scenarios.

There is also a section on Db2 12 considerations, as migration to Db2 12 requires that the BSDS has been converted to the extended RBA/LRSN format. Finally, we consider a very particular but rare case when operating in data sharing, where the BSDS of each member has been converted to extended format but you then have a requirement to disable data sharing. This is discussed in Db2 12 Considerations

If you are already in Db2 11 NFM and are preparing to migrate to Db2 12, then you should take into account the Db2 12 requirement that your BSDS is converted to the extended RBA/LRSN format before starting the migration. In fact, Db2 12 will not start if the BSDS has not been converted. Db2 12 has no requirement for you to convert the Db2 catalog and directory, or any application objects, to the extended RBA/LRSN format.

If you are non-data sharing, then you should consider the strategy of converting the objects before the BSDS to avoid the problem of speeding up the process of reaching the end of the RBA range. This is because converting the BSDS first will increase the size of all log records and accelerate your progress to the end of the log RBA range, increasing the risk of objects being placed in read-only status before you can convert them to the extended RBA. If this is a risk, do all the following while in Db2 11 NFM:

1. Reorg ALL the Db2 catalog and directory objects to convert them to the extended log RBA format
2. Set the OBJECT_CREATE and UTILITY_CONVERSION system parameters to EXTENDED
   a. New objects will be created in 10 byte extended format
   b. REORG, LOAD REPLACE and REBUILD will convert user objects to extended format without need to change utility control statements
3. Reorg all application objects to convert them to the extended LRBA/LRSN format, either as part of your normal REORG schedule, or by executing additional REORG jobs
   a. Regularly check your progress and identify unconverted objects by searching for rows with RBA_FORMAT = ‘B’ in the SYSIBM.SYxxxxPART catalog tables
4. Once all objects have been converted, set the OBJECT _CONVERSION system parameter to NOBASIC
5. Convert the BSDS.

If you are data sharing, you have less need to convert the Db2 objects, and can consider the following strategy:

1. In Db2 11 NFM, Convert the BSDS of ALL members of the data sharing group.
   a. Because converting the BSDS requires you to recycle the owning Db2 member, you can do this over a number of change slots, member by member.
2. At this stage, you have the option to migrate to Db2 12

The following steps are all optional and can be performed either in Db2 11 NFM or in Db2 12:
3. Reorg ALL the Db2 catalog and directory objects to convert them to the extended log RBA format
4. Set the OBJECT_CREATE and UTILITY_CONVERSION system parameters to EXTENDED
   a. New objects will be created in 10 byte extended format
   b. REORG, LOAD REPLACE and REBUILD will convert user objects to extended format without need to change utility control statements
5. Reorg all application objects to convert them to the extended LRBA/LRSN format, either
as part of your normal REORG schedule, or by executing additional REORG jobs

a. Regularly check your progress and identify unconverted objects by searching for rows with RBA_FORMAT = ‘B’ in the SYSIBM.SYSxxxxPART catalog tables

6. Once all objects have been converted, set the OBJECT _CONVERSION system parameter to NOBASIC

There are no special considerations for approaching the end of the log RBA range on one or more members in data sharing – you must convert the BSDS of all members to the extended format whilst in Db2 11 NFM, and then have the option to convert the Db2 catalog and directory and application objects. This can be done either in Db2 11 NFM or in Db2 12.

Disabling data sharing in a BSDS converted system

If, after reading this paper, you think that you are faced with special conditions which could affect your strategy, then it is recommended that you contact IBM Db2 for z/OS support for guidance.

**Non Data Sharing – approaching the end of the log RBA range**

<table>
<thead>
<tr>
<th>DB2 10 NFM</th>
<th>DB2 11 CM</th>
<th>DB2 11 NFM</th>
<th>Convert DB2 Objects</th>
<th>Convert BSDS</th>
</tr>
</thead>
</table>

**Data Sharing – approaching the end of the LRSN range**

<table>
<thead>
<tr>
<th>DB2 10 NFM</th>
<th>DB2 11 CM</th>
<th>DB2 11 NFM</th>
<th>Convert BSDS</th>
<th>Convert DB2 Objects</th>
</tr>
</thead>
</table>

**Data Sharing – approaching the end of log RBA range on one or more members**

<table>
<thead>
<tr>
<th>DB2 10 NFM</th>
<th>DB2 11 CM</th>
<th>DB2 11 NFM</th>
<th>Convert BSDS</th>
</tr>
</thead>
</table>

*Figure 17 Reaching the limits - conversion strategies*

**Non-data sharing – approaching the end of the log RBA range**

Non-data sharing Db2 environments require special considerations when converting to the extended RBA/LRSN, and the strategy is outlined in Figure 17 Reaching the limits - conversion strategies. Note that the high-level strategy is to convert the objects before converting the BSDS. This is because converting the BSDS first will increase the size of all log records and accelerate your progress to the end of the log RBA range, increasing the risk of objects being placed in read-only status before you can convert them to the extended RBA. It’s therefore important to convert the objects first, ensuring the task is completed before hitting the hard limit. It’s best to plan to complete well before hitting the hard limit, to allow for contingencies. The steps you should take, including migration from Db2 10, are as follows:
1. Run the pre-migration jobs and take any steps to clean-up the Db2 catalog and directory before starting migration.
2. Migration to Db2 11 CM
3. Migration to Db2 11 NFM
4. Reorg ALL the Db2 catalog and directory objects to convert them to the extended log RBA format
5. Set the OBJECT_CREATE and UTILITY_CONVERSION system parameters to EXTENDED
   a. New objects will be created in 10 byte extended format
   b. REORG, LOAD REPLACE and REBUILD will convert user objects to extended format without need to change utility control statements
6. Reorg all application objects to convert them to the extended LRBA/LRSN format, either as part of your normal REORG schedule, or by executing additional REORG jobs
   a. Regularly check your progress and identify unconverted objects by searching for rows with RBA_FORMAT = ‘B’ in the SYSIBM.SYSxxxxPART catalog tables
7. Once all objects have been converted, set the OBJECT_CONVERSION system parameter to NOBASIC
8. Convert the BSDS.

Data sharing – approaching the end of the LRSN range

The strategy for converting data sharing Db2 environments is also outlined in Figure 17 Reaching the limits - conversion strategies. Because Db2 object page headers record the LRSN and not the log RBA, there is no pressure to convert the objects before the BSDS. It is recommended in this case that you convert the BSDS of each member first, if only to avoid any diagnostic complications when running with some members with the BSDS in extended format and some members with the BSDS in basic format, alongside some objects in extended format and others in basic format, as discussed in Moving toward a true extended format Db2 subsystem. The recommended strategy is:

1. Run the pre-migration jobs and take any steps to clean-up the Db2 catalog and directory before starting migration.
2. Migration to Db2 11 CM
3. Migration to Db2 11 NFM
4. Convert the BSDS of ALL members of the data sharing group.
   a. Because converting the BSDS requires you to recycle the owning Db2 member, you can do this over a number of change slots, member by member.
5. Reorg ALL the Db2 catalog and directory objects to convert them to the extended log RBA format
6. Set the OBJECT_CREATE and UTILITY_CONVERSION system parameters to EXTENDED
   a. New objects will be created in 10 byte extended format
   b. REORG, LOAD REPLACE and REBUILD will convert user objects to extended format without need to change utility control statements
7. Reorg all application objects to convert them to the extended LRBA/LRSN format, either as part of your normal REORG schedule, or by executing additional REORG jobs
   a. Regularly check your progress and identify unconverted objects by searching for rows with RBA_FORMAT = ‘B’ in the SYSIBM.SYSxxxxPART catalog tables
8. Once all objects have been converted, set the OBJECT_CONVERSION system parameter to NOBASIC
Data sharing – approaching the end of the log RBA range on one or more members

The strategy for converting data sharing Db2 environments when the only challenge is that one or members are approaching the end of the log RBA range is again outlined in Figure 17 Reaching the limits - conversion strategies. Because Db2 object page headers record the LRSN and not the log RBA, and because you are not approaching the end of the LRSN range, there is no pressure to convert the objects at all. The only required activity is to convert the BSDS of each affected member. There is, however, no technical reason why you cannot go on to convert the catalog, directory and user page sets once you have performed BSDS conversion, with the proviso that we recommend that you convert the BSDS of all members prior to converting any objects, just as if you were approaching the end of the LRSN range. The recommended strategy in this case is:

1. Run the pre-migration jobs and take any steps to clean-up the Db2 catalog and directory before starting migration.
2. Migration to Db2 11 CM
3. Migration to Db2 11 NFM
4. Convert the BSDS of any members of the data sharing group approaching the end of the log RBA range.
   a. Because converting the BSDS requires you to recycle the owning Db2 member, you can do this over a number of change slots, member by member.

Db2 12 Considerations

If you are already in Db2 11 NFM and are preparing to migrate to Db2 12, then you should take into account the Db2 12 requirement that your BSDS is converted to the extended RBA/LRSN format before starting the migration. In fact, Db2 12 will not start if the BSDS has not been converted. Db2 12 has no requirement for you to convert the Db2 catalog and directory, or any application objects, to the extended RBA/LRSN format.

If you are non-data sharing, then you should consider the strategy of converting the objects before the BSDS to avoid the problem of speeding up the process of reaching the end of the RBA range. This is because converting the BSDS first will increase the size of all log records and accelerate your progress to the end of the log RBA range, increasing the risk of objects being placed in read-only status before you can convert them to the extended RBA. If this is a risk, do all the following while in Db2 11 NFM:

6. Reorg ALL the Db2 catalog and directory objects to convert them to the extended log RBA format
7. Set the OBJECT>Create and UTILITY>CONVERSION system parameters to EXTENDED
   a. New objects will be created in 10 byte extended format
   b. REORG, LOAD REPLACE and REBUILD will convert user objects to extended format without need to change utility control statements
8. Reorg all application objects to convert them to the extended LRBA/LRSN format, either as part of your normal REORG schedule, or by executing additional REORG jobs
   a. Regularly check your progress and identify unconverted objects by searching for rows with RBA_FORMAT = ‘B’ in the SYSIBM.SYSxxxxPART catalog tables
9. Once all objects have been converted, set the OBJECT _CONVERSION system parameter to NOBASIC
10. Convert the BSDS.

If you are data sharing, you have less need to convert the Db2 objects, and can consider the
following strategy:

7. In Db2 11 NFM, Convert the BSDS of ALL members of the data sharing group.
   a. Because converting the BSDS requires you to recycle the owning Db2 member, you can do this over a number of change slots, member by member.
8. At this stage, you have the option to migrate to Db2 12

The following steps are all optional and can be performed either in Db2 11 NFM or in Db2 12:

9. Reorg ALL the Db2 catalog and directory objects to convert them to the extended log RBA format
10. Set the OBJECT_CREATE and UTILITY_CONVERSION system parameters to EXTENDED
   a. New objects will be created in 10 byte extended format
   b. REORG, LOAD REPLACE and REBUILD will convert user objects to extended format without need to change utility control statements
11. Reorg all application objects to convert them to the extended LRBA/LRSN format, either as part of your normal REORG schedule, or by executing additional REORG jobs
   a. Regularly check your progress and identify unconverted objects by searching for rows with RBA_FORMAT = ‘B’ in the SYSIBM.SYSxxxxPART catalog tables
12. Once all objects have been converted, set the OBJECT_CONVERSION system parameter to NOBASIC

There are no special considerations for approaching the end of the log RBA range on one or more members in data sharing – you must convert the BSDS of all members to the extended format whilst in Db2 11 NFM, and then have the option to convert the Db2 catalog and directory and application objects. This can be done either in Db2 11 NFM or in Db2 12.

Disabling data sharing in a BSDS converted system

Disabling data sharing in a BSDS-converted system is an unusual scenario, but has been encountered by one customer who clones their production data sharing system to a non-data sharing pre-production environment on a quarterly basis. ‘BSDS-converted’ means that the BSDS of each member has been converted to extended format, but that some or all of the Db2 objects remain in basic format.

The reason this scenario is complicated is that disabling data sharing requires you to perform a cold start by creating a conditional restart record in the BSDS. Performing a cold start means that the log RBA values used to identify log records have to be set to a value higher than the highest written LRSN. With a basic format BSDS, if the next log record LRSN would have been D1485D778116, then the log RBA would also be set to D1485D778116.

However, with an extended format BSDS, an LSRN of 00D13B85F2725DA11000 would have to be converted to a log RBA of 00D13B85F2725DA11000. Since the log RBA is now greater than 6 bytes in length, NONE of the basic format table spaces can be updated. The only way round this to reorganize all the basic format table spaces specifying RBALRSN_CONVERSION EXTENDED. An optional mitigating strategy would be to convert all objects to extended format before disabling data sharing, but clearly your strategy should be determined by the requirements specified for your production system.

Considerations

When you convert a Db2 subsystem to support the extended RBA and LRSN format, you must keep in mind considerations for the RECOVER utility, disaster recovery, and other utilities.
The RECOVER utility
You use the RECOVER utility on a table space that is converted to the extended format with TOCOPY. If the image copy was taken when the table space was still in the basic format, the table space is reset to the basic format.

Disaster recovery
When you run any form of disaster recovery process, avoid any format conversion to or from the extended format during the recovery and rebuild process. Temporarily change the **UTILITY_OBJECT_CONVERSION** subsystem parameter to **NONE** (if not already the case) before the disaster recovery process starts. Also, ensure that you do not run Db2 utilities with the **RBALRSN_CONVERSION** keyword specified. Any format conversion increases the complexity and potentially the elapsed time of the recovery process. After you successfully complete the disaster recovery, change the **UTILITY_OBJECT_CONVERSION** setting back to its original value.

Other utilities
Some utilities have additional considerations. First, the **DSN1LOGP** and **REPORT RECOVER** utilities output RBA and LRSN values in the extended format. However, the values are still externalized in the basic format to any Db2 page sets that have not yet been converted to the extended format. Therefore, the **DSN1PRINT** utility outputs RBA and LRSN values in a different format to other Db2 utilities, when printing the contents of an unconverted object.

Use special care with the **DSN1COPY** utility. This utility does not perform any format conversion, for example from extended to basic format, nor does it update the catalog and directory information to reflect the actual format of the object.

However, if you are copying a table from one Db2 11 NFM subsystem to a Db2 11 subsystem in NFIM, ENFM or CM, you can use the **REPAIR** utility option, **REPAIR CATALOG TABLESPACE**, to ensure the catalog definition and the object format are consistent. This approach increases the number of steps and the elapsed time that are required for the copy process. The **REPAIR CATALOG TABLESPACE** option ensures consistency by updating the **RBA_FORMAT** column of the **SYSIBM.SYSTABLEPART** catalog table to reflect the actual format of the object in Db2 11 (Figure 21).

```
//S100 EXEC DSNUPROC,UID='CAT.REPAIR',
  // UTPROC='',SYSTEM='DBP1'
  //SYSIN DD *
  REPAIR CATALOG TABLESPACE DWDB.GLWSEMP
/*
```

**Figure 18** Db2 11 REPAIR CATALOG TABLESPACE

When an object is converted from the basic format to the extended format, you must avoid changing procedures. Therefore, when copying an object from one Db2 11 NFM subsystem to another, adopt one of the following approaches:

- Always run the **REPAIR CATALOG TABLESPACE** option to ensure that the catalog metadata reflects the physical format of the object.
- Run the **REPAIR CATALOG TABLESPACE** option with the **TEST** option. If the **TEST** option indicates that the catalog metadata is inconsistent with the physical format of the object, you must use the **REPAIR CATALOG TABLESPACE** option to update the **RBA_FORMAT** column of the **SYSIBM.SYSTABLEPART** catalog table to reflect the actual format of the object in Db2 11 (Figure 21).
object, run the **REPAIR CATALOG TABLESPACE** option without the **TEST** option.

The **REPAIR CATALOG TABLESPACE** utility option updates the catalog to reflect several physical characteristics, such as Reordered Row Format (RRF), Basic Row Format (BRF), Hash space value and so on. This option also indicates whether the object is in the extended or basic RBA or LRSN format. Before you run the **REPAIR CATALOG TABLESPACE** option, to understand all the implications of running the utility, see “REPAIR” ([ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.Db2z11.doc.ugref/src/tpc/Db2_z_utl_repair.dita](http://ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.Db2z11.doc.ugref/src/tpc/Db2_z_utl_repair.dita)).

However, you cannot run the **REPAIR CATALOG TABLESPACE** option to update the catalog if you are using the **DSN1COPY** utility to copy an object that was converted to the extended format from a Db2 11 NFM system to a Db2 10 NFM system. Even if you try to use the Db2 11 version of the **REPAIR** utility on a Db2 10 subsystem, the utility cannot succeed. The Db2 10 catalog does not contain the columns that were introduced in Db2 11 CM to support the extended format RBAs and LRSNs.

The following workarounds are available:

- Ensure that you migrate the target system to Db2 11 NFM before the source system. You can still use the **DSN1COPY** utility with the **REPAIR CATALOG TABLESPACE** utility to copy the data and to make sure that the catalog metadata is consistent with the physical format of the object.
- Ensure that any objects that you want to copy from one Db2 subsystem to another are not converted to the extended format until both systems are running at Db2 11 NFM.
- Use the **UNLOAD** utility to unload data from the source table or from an image copy, and then, use the **LOAD** utility to populate the target table. This option has the disadvantage of taking more elapsed time and consuming more CPU time than using the **DSN1COPY** utility. It is particularly unattractive if you have many tables or several large tables that you copy, for example, from one SAP landscape to another or from your OLTP system to your data warehouse.

Plan your Db2 11 migration strategy carefully. Take into account any requirement to copy data from one subsystem or data sharing group to another.

**Moving toward a true extended format Db2 subsystem**

With the flexibility of the conversion process, you might run in a mixed environment with objects in both the basic and extended RBA and LRSN formats for months or even years. Db2 11 can handle both formats concurrently and transparently.

However, you must be aware of the side effects that you might see when running in a mixed environment. For example, consider the data-sharing group, Db2G, with two members, Db2 M6 and Db2 M10. The BSDS is typically converted on a member-by-member basis. In this example, member M10 owns a converted BSDS and, therefore, creates the extended format of 10-byte log records. As you might have guessed from the naming convention, member M6 still has an unconverted BSDS and, therefore, creates log records in the basic format.

Now, imagine two transactions, one running on each member (Figure 22). They both access and update data in table spaces TSP6 (table GLWSEMP) and TSP10 (table GLWSDEP). Table space TSP6 is in the basic format. Therefore, the LRSN values are stored in the PGLOGRBA 41.
field of each page header (because it is a data sharing group). Table space TSP10 is in the extended format. Therefore, the LRSN values are stored in the PGBIGRBA field in the page trailer.

Figure 22 Two transactions accessing and updating data in two table spaces, each of which uses a different format

The first update occurs under control of member M10. The LRSN value that is used in internal Db2 processing is in the extended format. Because M10 also has an extended BSDS, the log records for this DML activity are also in the extended format. However, the table space that is being updated is in the basic format, and the associated PGLOGRBA field can only store 6 bytes. Therefore, Db2 must truncate the internal LRSN value ‘00100000000001000002’X and store it in the PGLOGRBA field as ‘100000000001’X.

The next update also occurs under control of member M10, but this time on table space TSP10, which was already converted to the extended format. Therefore, Db2 does not truncate the internal LRSN value. Instead, it stores the 10-byte value of ‘00100000000001000003’X in the Db2 11 PGBIGRBA page trailer field.

Now consider what happens when update activity occurs on member M6. Although Db2 works internally with log records in the extended format, M6 still writes basic format log records to the active logs because its BSDS is not yet converted. In this case, a data change occurs at ‘00100000000001000004’X.

Because the updated table space is TSP10, a complete 10-byte LRSN value is stored in PGBIGRBA. However, Db2 stores a value of ‘00100000000002000000’X in PGBIGRBA, although the original LRSN value is ‘00100000000001000004’X. The reason is that Db2 must ensure that the LRSN value is greater than the previous LRSN value that was written to the page and recorded in PGBIGRBA. Because member M6 still writes log records in the basic format.
format to the active log, the next valid LRSN value must be greater than the equivalent 6-byte value of the 10-byte LRSN value that was previously stored in PGBIGRBA. The next valid 6-byte LRSN value is ‘100000000002’X, which must then be expanded again and stored as ‘001000000002000000’X in PGBIGRBA.

The next update driven by member M6 is to the basic format table space, TSP6, and occurs at LRSN ‘00100000000001000005’X. Db2 has to truncate the LRSN value to 6 bytes and, therefore in this case, the TSP6 page header field PGLOGRBA contains ‘100000000002’X.

For the fifth update, which is to TSP6, member M10 must again generate a larger LRSN value. But, because the equivalent 6-byte value of the M10 internal LRSN, ‘100000000002’X, is still not greater than the existing LRSN value in PGLOGRBA, M10 informs the Db2 log manager that the next valid LRSN value must be greater than ‘001000000002FFFFFF’X. That is,Db2 M10 might have to wait up to 16 microseconds before a large enough LRSN value is returned by the Sysplex Timer.

For the sixth and final update in our example, Db2 M10 can safely store the internal LRSN value of ‘00100000000200000003’X in the PGBIGRBA field of TSP10 without any modification.

Figure 23 illustrates the final phase of conversion to a true, 10-byte Db2 data sharing group. This phase is achieved when you convert the BSDS and all user and catalog objects. All logging is in the extended format, and all pages in the user and catalog objects record 10-byte LRSN or RBA values. Now, all possibility of LRSN spin is eliminated. Also, you will not exhaust the logging capacity of Db2 unless you exceed 1 YB of log data (keeping in mind that Db2 10 provided 256 TB of log capacity) or run the system for over 35,000 years.
Conclusion

Use of the extended RBA and LRSN is already a reality for customers who have migrated to Db2 11 NFM and are in the process of converting their Db2 objects to the extended RBA and LRSN. At the time of writing this white paper, one customer already reported completing the entire process in a production environment, converting over 250,000 user objects, including tables and indexes, in addition to the BSDS and the recovery log.

Customer experiences so far have been positive and confirm that the migration process is robust and sound. This message will hopefully give you the confidence that, if you plan thoroughly and carefully, taking into account the guidance described in this white paper, you can look forward to a successful conversion project.

Data sharing customers who have embarked on this journey and customers who have completed the extended RBA and LRSN conversion have all split the RBA and LRSN project into two distinct phases, even though this approach is not strictly required. Although you have total flexibility in the order in which these phases occur, focus the first phase on the Db2 subsystem itself, that is, on the BSDS, the catalog and the directory. Focus the second phase, which is typically the bigger task, on converting all Db2 user objects to the extended RBA and LRSN.

Regardless of how you approach your project, you are now armed with a solid grounding in the conversion process. You also now have a good understanding of the technical details to ensure that you can move to the extended RBA and LRSN format safely and successfully.

For more information

For more information about Db2 11 z/OS, see ibm.com/software/data/Db2/zos.

In addition, see the following technical references:

- “The expanded RBA and LRSN in Db2 11 for z/OS” technote at https://ibm.biz/BdE5nf
- “Resetting the log RBA value in a data sharing environment” in the IBM Knowledge Center for Db2 10 for z/OS (ibm.com/support/knowledgecenter/SSEPEK_10.0.0/com.ibm.Db2z10.doc.admin/src/tpc/Db2z_subsystemdatashrrba.dita)
- “Resetting the log RBA value in a non-data sharing environment” in the IBM Knowledge Center for Db2 10 for z/OS (ibm.com/support/knowledgecenter/SSEPEK_10.0.0/com.ibm.Db2z10.doc.admin/src/tpc/Db2z_subsystemendrba.dita)

About the authors

John Campbell is a Db2 for z/OS Distinguished Engineer and the founder of the Db2 for z/OS SWAT Team.

Michael Dewert is a Consulting IT Specialist and a member of the Db2 for z/OS SWAT Team.
Gareth Jones is an IT Specialist and a member of the Db2 for z/OS SWAT Team.

Acknowledgments

A special thank you to Timm Zimmermann of Rocket Software who assisted with this paper.

Appendix A. Converting a 10-byte LRSN to a time stamp

With Db2 10 for z/OS and earlier versions, you can use the built-in TIMESTAMP function to convert any 6-byte LRSN value to a timestamp. For example, a 6-byte LRSN value of ‘CBE2B5955DCF’X can be converted to a timestamp (Figure 24).

```
SELECT TIMESTAMP('CBE2B5955DCF' || '0000') FROM SYSIBM.SYSDUMMY1;
```

Figure 24 A 6-byte LRSN value converted to a timestamp

In Db2 11 CM mode, you can still use the TIMESTAMP function, but with two possible input formats, the 6-byte and 10-byte formats.

In general, 6-byte LRSN values are in Db2 11 CM in the Db2 recovery log, in the catalog and directory, and in data pages. As in the previous example, 6-byte LRSN values can be copied and pasted to the TIMESTAMP function.

However, all SYSLOG and utility messages report 10-byte LRSN values, except for the DSN1PRNT utility. A different copy and paste technique is needed for these values. The leading byte and the trailing 3 bytes of a 10-byte LRSN value should be excluded. If you prefer to count in hexadecimal digits, the first two digits must be excluded, and only digits 3 - 14 must be used as input (Figure 25).

```
10 byte LRSN value = '00CBE2B5955DCF086C00'X

SELECT TIMESTAMP('CBE2B5955DCF' || '0000') FROM SYSIBM.SYSDUMMY1;
```

Figure 25 Hexadecimal version shown, excluding the first two digits and using only digits 3 - 14

With Db2 11 NFM, the 10-byte format is used in all Catalog and Directory RBA and LRSN columns, and in all messages. The 6-byte format is still used in unconverted data pages, with the DSN1PRNT utility reporting 6-byte values for these pages. You can still use the built-in TIMESTAMP function, but be aware that conversion is valid only if the leading two digits are x’00’.

```
Important: Use this technique only if the first two digits are 00. Otherwise, the conversion is not usable.
```

As of Db2 11 NFM, you can also use the binary hex modifier, BX. The 6-byte values can be copied by using cut and paste and padding the right side with 0000 (Figure 26). The 10-byte
LRSN values can also be used, if the first two digits are cut and digits 3 - 18 are used. To use the binary hex function, not only must you be using NFM, but the package used, whether it is DSNTEP2 or one of the REXX packages, must be bound with the APPLCOMPAT(V11R1) specification.

```
SELECT TIMESTAMP(bx'CBE2B5955DCF0000') FROM SYSIBM.SYSDUMMY1;
```

**Figure 26** Binary hex modifier with a 6-byte value, padded with 0000

**Important:** Use this technique only if the first two digits are 00. Otherwise, the conversion is usable.

If you find that you must be able to convert an LRSN to a timestamp where the first two digits are x‘00’, in Db2 11 NFM, you can still use the data type modifier bx, by padding the right side with 000000000000. For example, for a 10-byte LRSN value of ‘01CBE2B5955DCF086C00’, you can use the query shown in Figure 27.

```
SELECT TIMESTAMP(bx'01CBE2B5955DCF086C00000000000000') FROM SYSIBM.SYSDUMMY1;
```

**Figure 27** Query for converting a 10-byte LRSN value to a timestamp

You can also use this form of query for 6-byte values, by using cut and paste and padding the value with 00 on the left side and 000000000000000000 on the right side.