

Same Input, Same Output

Proving that a sort moved off the mainframe is faithful: what “identical” means, where it breaks, and how to test for it

Bill Ahlbrandt

President and CEO, Ahlbrandt Software

Abstract

When a batch workload moves from z/OS to Linux, Windows or the cloud, everybody expects its sort steps to produce the same output they always did. This paper looks at what “the same” has to mean for a sort. The records and their order are only the start. Record counts, record lengths, header and trailer records, printed reports, return codes and the messages your operations tooling reads all count too. Records with equal keys are the most common reason two correct sorts disagree, and if the original runs without order preservation there may be no single right answer to compare against. After that I lay out a test regime: production captures, byte-level hashing, record-count reconciliation, full comparison with no sampling, and a stretch of parallel running. The last section puts the cost of getting this wrong next to some documented failures in UK banking.

What I found

- A sort has at least **eight outputs you can observe**. A migration can keep the data right and still change the return code, the record lengths or a printed report, so each one needs its own test.
- Records with equal keys can legitimately come out in **any order** unless order preservation is on. IBM documents DFSORT's shipped default as preserving it only for variable-length records handled by Blockset. Syncsort MFX defaults to not preserving it.
- Duplicate removal and SUM keep **one record from each group**, and which one survives depends on that same order. The totals can reconcile while the surviving record, and every field that isn't summed, comes out different.
- Sampling is a weak test for a sort. If a defect touches one record in 100,000, a random sample of 10,000 records has **less than a one-in-ten chance** of catching it. Hashing makes it cheap to compare everything.
- None of the documented banking failures of the past decade came from a sort. They came from testing and change control, and they show what regulators expect a firm to have proved before it changes a production platform.

1. What has to be proved

A sort step reads a file, rearranges it (and maybe filters, reformats or totals the records), and writes files that later steps read. When the batch stream around it moves to a new platform, everyone expects the new sort to produce what the old one did, and nothing downstream to notice. That's a reasonable thing to expect, because a sort is described by control statements that say what result you want. It's still a claim, though, and in a regulated business a claim about production data has to be proved. In this paper I go through what that takes.

It's harder than it sounds. For one thing, a sort produces more than one data set. Sort specifications are also incomplete on purpose. They say how to order records by key and often say nothing about records whose keys are equal. And the evidence usually comes from sampling, even though the defects most likely to matter in a sort are exactly the ones a sample is least likely to find.

2. What “identical” means for a sort

The narrowest definition is that the main output file has the same bytes in the same order. You need that, but it isn't enough. The job scheduler reads the step's return code. It never looks at SORTOUT. Operations reads the record counts in the job log, and a downstream program may check a trailer record before it reads anything else. Exhibit 1 lists everything a sort step produces and how each piece can change.

Exhibit 1. Everything a sort step produces

Output	What must match	How it can differ after migration
Record content	Every byte of every output record	Character or numeric conversion; padding with spaces instead of binary zeros; reformatting differences
Record order	Sequence of records, including within equal keys	Equal keys handled differently; a different collating sequence
Record count	Records read, written and dropped, per output file	Filtering or duplicate removal picking different records; short or incomplete records handled differently
Record length	Fixed length, or each variable length and its descriptor	Record descriptor words kept, stripped or rebuilt; trailing blanks trimmed
Headers and trailers	Counts, totals, dates in control records	Totals formatted differently; run dates generated on the new platform
Reports	Every printed line, including carriage control	Page breaks, carriage-control bytes, edit masks, date and time stamps
Return code	Condition code for each step	Warnings mapped to a different code; a condition treated as an error on one side only
Messages	Log lines that automation reads	Different message IDs, wording or record-count lines

Source: author's analysis. Applies to any sort product on any platform.

Return codes need special care, because they decide what happens next. DFSORT lets a shop decide whether a condition is ignored, flagged or fatal. When a summed field overflows, OVFL0=RC0 carries on with a return code of 0, RC4 carries on with 4, and RC16 stops with 16. There are similar choices for an empty output file (NULLOUT), for output records longer or shorter than the input (PAD and TRUNC) and for incomplete spanned records (SPANINC).¹ ICETOOL's COUNT operator is there to set a non-zero return code when a record count meets a condition, so the JCL can skip or run later steps.² So a replacement sort that writes the right records but returns 0 where the original returned 4 has changed how the batch schedule behaves, even though every output file compares equal.

Record lengths, RDWs and file transfer

Variable-length records on z/OS carry a four-byte record descriptor word (RDW) that holds the record's length. Whether it survives a transfer depends on the tool. IBM's own support note says the same VB data set arrives with four extra bytes on each record when it's sent by Connect:Direct in binary, and without them when it's sent by FTP in binary, which strips the RDWs unless you tell it to keep them.^{3, 4} Neither one is wrong. But a comparison that doesn't know which convention each side used will report every record as different, or worse, somebody will adjust it until it reports nothing. So pick the transfer method as part of the test design.

Character encoding, collating sequence and numeric formats are the other big source of differences. DFSORT collates character data in EBCDIC or ASCII, whichever you specify,⁵ and the two put digits, letters and punctuation in different orders. Packed and zoned decimal fields carry signs you can't read off the bytes without knowing the field type. A later paper in this series covers these. Settle them before fidelity testing starts, because a test can't tell an encoding decision from a defect.

3. Equal keys: where two correct sorts disagree

When two records have identical values in every control field, the sort specification doesn't say which comes first unless it asks for order preservation. IBM's manual is explicit: such records "retain their original input order or are arranged randomly, depending upon which of the two options, EQUALS or NOEQUALS, is in effect."⁵ A sort that keeps equal records in input order is called *stable*. A lot of general-purpose sorts aren't. The C standard leaves the order of equal elements under *qsort* unspecified,⁶ and GNU *sort*, the default on most Linux systems, breaks ties by comparing whole lines unless you give it `--stable`.⁷

The defaults matter because most production control statements don't say either way. IBM's informational APAR II09491 documents the shipped DFSORT default as EQUALS=VLBLKSET. Order is preserved for variable-length records processed by DFSORT's Blockset technique, and doesn't have to be preserved for anything else.⁸ Syncsort MFX says NOEQUALS is its default.⁹ Either one can be overridden by the installation or in the job, so you have to find out which setting each step actually ran with. Don't assume it. Micro Focus's MFSORT, on the other hand, was described in a 2013 answer on the vendor's forum as always returning duplicates in input order.¹⁰

Here's the uncomfortable part. If the original step ran with NOEQUALS, its production output is one of many valid results, and a faithful replacement may legitimately produce a different one. IBM warns that EQUALS can slow things down when it isn't needed,⁸ so NOEQUALS is common. A byte comparison will flag the difference. What you really need to know is whether anything downstream depends on the order, and a lot of the time something does.

When the order of equal keys turns into data

Order within equal keys turns into data whenever a later operation picks records by position. DFSORT's SUM statement collapses each group of equal-keyed records into one. IBM's tutorial documentation says that "when summing records keeping the original order, DFSORT chooses the first record to contain the sum";¹¹ Syncsort documents the same rule under EQUALS.⁹ Without order preservation, a member of IBM's DFSORT development team put it plainly in 2008: "If NOEQUALS is in effect, it's random."¹² The summed fields will reconcile, but every field that isn't summed comes from whichever record survived. The same goes for sequence numbers added after the sort, output split across files in rotation, picking the first or last record for each key, and keeping only the first *n* records. Exhibit 2 shows what happens to five records.

So here's the practical rule. For every sort step, write down whether equal-key order was guaranteed on the mainframe. If it was, the replacement has to reproduce it and you can compare byte for byte. If it wasn't, one of two things is true. Either the business owner agrees that order within a key doesn't matter, and you sort both outputs on the full record before comparing them. Or the step does depend on it, and the original job has a latent bug the migration just exposed. Adding EQUALS on both sides and taking a new baseline is usually the cleanest fix, and after that every comparison is deterministic.

Exhibit 2. Five records, one sort key, three correct answers (hypothetical)

Input (key, type, amount)	EQUALS	Whole-line tie-break	Another valid NOEQUALS order
a 2002 CR 050	b 1001 DR 120	d 1001 CR 030	b 1001 DR 120
b 1001 DR 120	d 1001 CR 030	b 1001 DR 120	d 1001 CR 030
c 2002 DR 075	a 2002 CR 050	e 2002 CR 010	c 2002 DR 075
d 1001 CR 030	c 2002 DR 075	a 2002 CR 050	e 2002 CR 010
e 2002 CR 010	e 2002 CR 010	c 2002 DR 075	a 2002 CR 050
After SUM on amount:	1001 DR 150	1001 CR 150	1001 DR 150
	2002 CR 135	2002 CR 135	2002 DR 135

Control statements: SORT FIELDS=(1,4,CH,A) then SUM FIELDS=(9,3,ZD). The letters identify input records and aren't part of the data. All three orders satisfy the sort specification. With 2 and 3 records per key there are $2 \times 6 = 12$ valid orders. The “whole-line tie-break” column is the order GNU sort without --stable would give if asked to sort on the key alone. The totals per key agree in every case. The debit/credit flag on the surviving record doesn't.

4. Capture, replay, compare

Public write-ups of mainframe migration testing all follow the same basic pattern. AWS's guidance is to run the same tests on source and target with identical inputs and compare the results.¹³ Its Application Testing service, generally available since June 2024, captures reference data sets on the mainframe, replays the workload on the target and classifies each record as identical, equivalent, different or missing.^{14,15} “Equivalent” is governed by explicit rules (for example, for dates relative to the run date), so expected differences have to be written down up front. Nobody gets to wave them through.¹⁴ For sort steps the method has six stages.

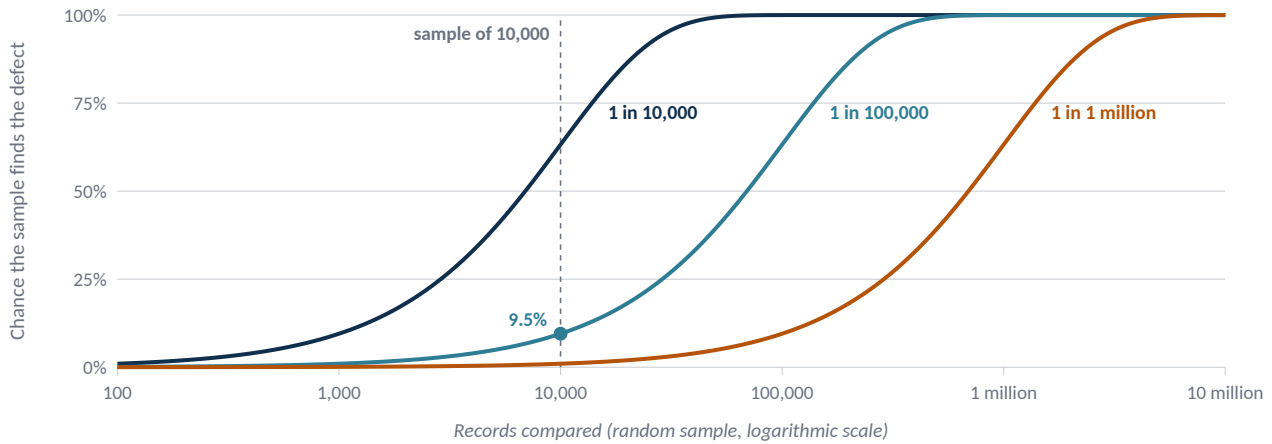
- 1 Inventory and classify.** List every sort, merge, copy and ICETOOL step with its control statements, its effective options (EQUALS above all), its record formats and the return codes it has produced in production. DFSORT's SMF type 16 records give you record counts for each invocation.¹⁶
- 2 Capture on z/OS.** For a business cycle you pick, keep each step's inputs and outputs, its return code and its message log. Transfer in binary, with a written-down convention for RDWs. Never send it through a text-mode conversion.
- 3 Fingerprint both sides.** Compute a cryptographic hash such as SHA-256¹⁷ of every output, on the same platform with the same tool, and record the record count and the length profile (shortest, longest, distribution).
- 4 Replay on the target.** Run the migrated step against the captured inputs, with the same run date and parameters, and fingerprint its outputs the same way.
- 5 Compare, then explain.** If the hashes match, that file is done. If they don't, run a record-level comparison that reports the first difference, its offset and field, and whether records are missing, extra, reordered within a key or changed.
- 6 Reconcile what isn't in the file.** Match return codes step by step. Match record counts against the mainframe's log and SMF data. Compare reports line by line, with date, time and page fields masked by a written rule and the carriage-control bytes left in.

Reports are worth the extra effort. DFSORT puts an ANSI carriage-control character in the first byte of every OUTFIL report line unless you specify REMOVECC,¹⁸ so a comparison that strips it will hide page-break differences. Keep the masks for run dates and times narrow. A rule that ignores a whole header line also ignores a wrong record count printed on that line.

5. Sampling, regression and parallel running

Full comparison sometimes gets written off as too expensive, and somebody checks a sample instead. For sort that's a bad trade. The defects that matter tend to be rare and conditional: a record at the maximum length, a sign nibble that only shows up on reversals, a key group with more than one member. Exhibit 3 shows the arithmetic. If a defect affects one record in 100,000, a random sample of 10,000 records finds it less than one time in ten.

Exhibit 3. Why sampling rarely catches rare defects (illustrative)



Lines show the share of records affected by a defect in a 10-million-record output file.

Probability that a simple random sample contains at least one affected record, $1 - (1 - p)^n$, approximated as $1 - e^{-pn}$. Assumes the affected records are scattered at random. An ordering defect within equal keys can't be seen by any sample that matches records by key, however big it is.

Hashing takes care of the cost argument. A SHA-256 digest is one sequential pass over each file, so full comparison costs about as much as reading the output one more time, and when the hashes agree you're finished. A sample is useful for explaining differences a full comparison has already found. It can't tell you there aren't any.

Keep it as a regression suite

The captured inputs, expected outputs and fingerprints make a regression suite that outlives the migration. Any later change to the sort product, operating system, locale or job can be re-run against it. Put in the cases an ordinary day doesn't exercise: empty input, a single record, maximum record length, short variable-length records, sums that overflow, keys with lots of duplicates, and month-end, quarter-end and year-end.

Parallel running

Replay proves the steps you captured. Parallel running proves the whole cycle. Google Cloud's Dual Run, announced in March 2023, keeps the mainframe as the primary system while the migrated workload runs as a secondary on the new platform. Results from both are validated and differences reported before any traffic is switched.¹⁹ Google's own white paper suggests this phase typically runs six months to a year.²⁰ Let the calendar set the length of the parallel run. A count of clean days isn't enough. If it hasn't been through a month-end, and ideally a quarter-end, it hasn't seen the jobs most likely to produce a regulatory figure. Exhibit 4 pulls the checks together.

Exhibit 4. A fidelity checklist for sort steps

Check	Method	Pass criterion
1. Output bytes	SHA-256 of each output data set, computed on one platform with one tool	Hashes equal, or every difference explained and signed off
2. Record counts	Input, output and per-OUTFIL counts against the mainframe log and SMF type 16	Exact match for every file
3. Record lengths	Shortest, longest and length distribution; RDW convention written down	Identical profile
4. Equal-key order	Effective EQUALS setting per step; if order isn't guaranteed, sort both sides on the full record and compare	Byte match where guaranteed; owner sign-off where not
5. Survivors	SUM, duplicate removal, first or last per key: compare the fields that aren't summed	Same record survives for every key
6. Headers, trailers	Field-by-field comparison of control records	Counts and totals equal; run dates masked by written rule
7. Reports	Line by line, carriage control kept; narrow masks for date, time and page	No unmasked differences
8. Return codes	Step-by-step match, including forced warnings (empty input, overflow, truncation)	Same code for every case
9. Messages	Lines read by automation or operators	Same IDs and counts, or automation rules updated and tested
10. Parallel run	Daily comparison of checks 1–9 across full business cycles	Clean through at least one month-end, ideally a quarter-end

Source: author's synthesis of the practices in this paper and in AWS and Google Cloud public guidance.^{13, 14, 19}

6. The cost of getting it wrong

No public incident report blames a major banking failure on a sort. The documented failures still matter. They show what happens when a change to a production platform goes live without being proved, and what regulators expect afterward.

In April 2018 TSB migrated the data for its corporate and customer services onto a new platform, and a significant share of its 5.2 million customers were hit by the disruption that followed.²¹ The independent review by Slaughter and May, published by TSB's board in November 2019, found that parts of the two data centers built for the new platform had been configured inconsistently, and that testing hadn't caught it.²² The migration of the customer data itself was reported as complete.²² TSB recognized £330.2 million of post-migration costs in 2018,²³ and in December 2022 the FCA and PRA fined it £48.65 million in total.²¹ For a sort migration the lesson is the same on a smaller scale. Environments that were supposed to behave identically didn't, and only a test designed to find the difference would have found it.

Before that, in June 2012, an upgrade to the software that processed overnight account updates at RBS, NatWest and Ulster Bank affected more than 6.5 million UK customers. The regulators' fines came to £56 million, and the FCA listed weaknesses in testing among the causes.²⁴ In regulatory reporting, the FCA fined Goldman Sachs International £34.3 million in 2019 over 220.2 million transaction reports that were incomplete, inaccurate or shouldn't have been made. It cited weaknesses in change management and in reconciling reports against the firm's own records.²⁵ None of these was caused by a sort. What they show is that the burden of proof is on the firm, and regulators expect reconciliation to be systematic.

7. Bottom line

- **Define identical before you test for it.** Content, order, counts, lengths, control records, reports, return codes and messages are separate outputs. Each one needs a check and a pass criterion.
- **Find out the equal-key rule for every step.** If the original didn't preserve order, there's no single right answer. Decide whether order matters. If it does, fix the job and leave the comparison alone.
- **Compare everything.** Hashes make full comparison cheap, and the defects that matter in sort are the rare ones a sample misses.
- **Run in parallel through the calendar.** Month-end and quarter-end are when the reports regulators read get produced. If your parallel run hasn't seen them, you aren't done.

About the author

Bill Ahlbrandt is President and CEO of Ahlbrandt Software, which he founded in 1993. Before that he was a systems programmer, Director of Information Security and Director of R&D at Acxiom. He has been working on mainframe sort and batch for more than thirty years, and his company's sort engine runs mainframe sort jobs on Linux and other platforms for banks, insurers and government agencies.

Full disclosure: my company sells sort software for platforms other than the mainframe, built to reproduce mainframe sort output. The test methods in this paper apply to any replacement, including mine.

References

1. IBM, *z/OS DFSORT Application Programming Guide* (V2R2), "Specifying EXEC/DFSPARM PARM options": NULLOUT, OVFO, PAD, TRUNC, SPANINC.
2. IBM, *z/OS DFSORT Application Programming Guide*, ICETOOL COUNT operator.
3. IBM Support, "When C:D z/OS transfers a VB file there are 4 extra bytes at the beginning of the record that are not there when transferring the same VB file with FTP", technote.
4. IBM, *z/OS Communications Server IP Configuration Reference* (V2R1), "RDW (FTP client and server) statement".
5. IBM, *z/OS DFSORT Application Programming Guide* (V2R1), "Control fields and collating sequences".
6. ISO/IEC 9899:2011, *Programming languages — C*, §7.22.5.2, the qsort function.
7. Free Software Foundation, *GNU Coreutils manual*, "sort invocation".
8. IBM, informational APAR II09491 (DFSORT EQUALS/NOEQUALS support, APAR PN71337), last modified 19 August 1998.
9. Precisely, *Syncsort MFX 3.1 Programmer's Guide*, "EQUALS/NOEQUALS Parameter". Vendor documentation.
10. Rocket Software Community forum, answer on MFSORT stable ordering, 15 February 2013. Vendor forum statement.
11. IBM, *Getting Started with DFSORT*, Release 14, SC26-4109-08, p. 28.
12. F. Yaeger, IBM DFSORT development, IBM-MAIN mailing list, "DFSORT question MERGE w/SUM FIELDS=NONE", 15 January 2008.
13. AWS Prescriptive Guidance, "Testing", in *Replatforming mainframe applications with a shared Db2 database*.
14. AWS, *AWS Mainframe Modernization User Guide*, "AWS Mainframe Modernization Application Testing concepts".
15. AWS, "AWS Mainframe Modernization Application Testing is now generally available", 12 June 2024.
16. IBM, DFSORT SMF type 16 record documentation and reporting samples, 15 August 2022.
17. NIST, FIPS PUB 180-4, *Secure Hash Standard*, August 2015.
18. IBM, *DFSORT: Ask Professor Sort*, DFSORT web site document (OUTFIL reports and REMOVECC).
19. A. Gomathinayagam and T. Nikl, "Dual Run by Google Cloud helps mitigate mainframe migration risks", Google Cloud blog, 16 March 2023.
20. Google Cloud, *Dual Run: Replatforming solution*, white paper, 13 July 2023.
21. Financial Conduct Authority, "TSB fined £48.65m for operational resilience failings", press release, 20 December 2022.
22. TSB Bank, "TSB Board publishes independent review of 2018 IT Migration" (Slaughter and May review), 19 November 2019.
23. TSB Bank, "TSB announces 2018 full year results", 1 February 2019.
24. Financial Conduct Authority, "FCA fines RBS, NatWest and Ulster Bank Ltd £42 million for IT failures", 20 November 2014; PRA fine of £14 million announced the same day.
25. Financial Conduct Authority, press release and Final Notice, Goldman Sachs International, 28 March 2019.