The Attachment Problem: What a Signed Record Should Hold

A signed record that points at a file on a shared drive is a signed pointer. Backups protect you from losing the file. Almost nothing protects you from someone quietly changing it.

Brandon Holland, Marketing & Operations Digital Solutions Architect

September 3, 2026 · 9 min read

IN BRIEF

A signed electronic record should hold its own evidence. Drafts can reference live files, so corrections flow in, but at signing the evidence should be captured into the record, provably unchanged from then on, with the record able to say if the original has since moved. Backups prevent loss. Only capture at signing prevents silent change.

Key takeaways

- A backup answers whether the file is still there. It never answers whether it is still the same file.
- A record that references a file it does not hold has signed a pointer. The signature covers the path, not the file.
- Capture the evidence into the record at signing, and let drafts track the live file until then so corrections flow in.
- A record should be able to say the source changed after approval. That is what labs currently find out too late.
- A checksum and a trusted timestamp prove different things. Ask a vendor which one they mean when they say tamper-evident.

Ask a laboratory how it protects its electronic records and the answer is a backup schedule. It answers a question about losing files. It is no answer at all to the question an inspector actually asks: whether the file behind a signed result is the file that was there when it was signed.

Most laboratory records are not self-contained. The result in the notebook or the LIMS points at a chromatogram, a scanned certificate or an instrument export that lives somewhere else: a shared drive, an instrument PC, a document store. The signature covers the record. It does not cover the file, and the file can be replaced by anyone with access to the folder, leaving the signed record pointing at something the signatory never saw. This article is about that gap and what a signed record should hold instead. It sits beside the site's guides to electronic signatures and Part 11, and asks a narrower question than either.

What is the difference between losing a file and the file changing?

Losing a file is loud. The link breaks, the folder is empty, somebody notices within a day and the backup is restored. It is disruptive, it is embarrassing in front of an auditor, and it is recoverable, because the whole apparatus of backup and restore was built for exactly this. Every lab has a story about a lost drive and almost every story ends with the data back.

A file changing is silent. The link still works. The folder still has a file of that name. The record still points at it, the signature is still valid, and nothing anywhere indicates that the file is not the one that was there on the day. It might have been re-exported from the instrument with different integration parameters. It might have been opened, "tidied" and saved. It might have been overwritten by a later run with the same filename. Nobody notices, because nothing is missing.

The two failures need different defences, and the second is the one regulators care about. A backup answers "is the file still there". It never answers "is it still the same file", because the backup faithfully preserves whatever the file has become. Data integrity guidance from every major regulator turns on the record being complete, consistent and enduring, and a record whose evidence can change after approval is none of those, however many copies of it exist.

Why is a signed pointer not a signed record?

An electronic signature binds a person to the content of a record at a moment. That is its whole meaning: this person stood behind this content, then. If the content is a reference to a file elsewhere, the signature

binds the person to the reference, not to the file. What they stood behind was a path. What the path leads to today is whatever is there today.

Laboratories rarely see this as a problem because the two are usually the same: the chromatogram on the drive on the day of signing is, in most cases, still there a year later. But “usually” is not what a signature is for. The case that matters is the one where the file was changed, deliberately or by accident, and the signed record gives no sign of it. In that case the lab has a result with a valid signature, a reviewer who approved it, and evidence that no longer supports it, and the lab does not know.

> NOTE
>
> The problem is not confined to shared drives. A record that links to a file in a document management system, a cloud folder or an instrument’s own data store is a signed pointer too, unless the target is locked at the moment of signing and the record can prove it has not changed. The question to ask of any system is not where the file lives but what the record holds.

The regulatory framing is direct. The rules on electronic records require that they be protected to enable accurate and ready retrieval throughout the retention period, and that the signature be linked to the record so it cannot be excised, copied or transferred to falsify a record. A signed record whose supporting file can be silently replaced meets neither requirement in substance.

What should happen to referenced files at the moment of signing?

While work is in progress, a record should be allowed to reference live files, because corrections should flow in. An analyst re-processes a run, the instrument writes a corrected export, and the draft record should see the corrected file without anyone re-attaching it. Freezing evidence too early produces records that are complete, signed and pointing at a superseded file.

At signing, that changes. The record is finished, the person is standing behind it, and from that moment the evidence should be part of the record rather than a reference to something outside it. The referenced files should be captured into the record itself, stored with it, and given a checksum at capture so that the record

can later prove the file is exactly what it was on the day. The record then holds its own evidence and stops tracking the source. That is the entire point of signing.

Two edge cases decide whether the approach is honest. A file too large to carry should be referenced rather than copied, and the record should say so visibly, so that a signed record with a two-gigabyte raw data set outside it is never mistaken for one that holds everything. And a capture that fails, because the source was unreachable at the moment of signing, should leave the record saying the capture failed. Worse than a missing attachment is a signed record that appears to hold evidence it does not.

- Draft records reference live files, so a corrected export is seen without re-attaching
- At signing, every referenced file is captured into the record and stored with it
- Each captured file receives a checksum at capture, kept with the record
- A file too large to capture is referenced instead, and the record says so on its face
- A failed capture is recorded as failed; the record never looks complete when it is not
- After signing, the record stops tracking the source and can report whether the source has since changed
- Public or anonymous sharing of the files behind records is off unless someone turns it on for a stated reason

How can a record tell you the source has moved?

Once the record holds a checksum of the file as it was at signing, a second question becomes answerable that was not before. The source file, still sitting on the shared drive or in the document store, can be checked against that checksum at any time. If they match, the source is what it was. If they do not, the source has changed since the record was approved, and the record can say so.

That warning is exactly what laboratories currently find out too late. Today, the discovery that a file behind an approved result has been altered comes during an investigation, months later, when someone compares a printout in a binder against the file on the drive and finds they differ. By then the questions are about who changed it and when, and the answers are usually unavailable. A record that can check its own source turns that into a routine finding, raised when the drift happens rather than when it is discovered, and the investigation starts with the fact rather than the suspicion.

It is worth being precise about what the warning means. It does not mean the signed record is compromised; the record holds its own copy and that copy is unchanged. It means the world outside the record has diverged from it, which is information the quality unit wants. Perhaps the source was legitimately re-processed and a new record should be raised. Perhaps it was tidied by someone who should not have touched it. Either way, the lab now knows, and knowing is the whole difference. The document store that holds the source and the record that holds the copy are doing different jobs, and the warning is the seam between them made visible.

Where does a checksum stop and a timestamp begin?

A checksum proves one thing: the file has not changed since the checksum was taken. It proves that to anyone who trusts the system that took it. If the laboratory itself is the party under suspicion, because an inspector is asking whether the lab altered its own records and then re-computed the checksums, the checksum proves nothing, because the lab controls both.

That is what a trusted timestamp is for. An independent third party attests that a particular checksum existed at a particular time, and the lab cannot backdate or alter that attestation because it does not hold the key. A checksum answers “has this changed since we stored it”. A third-party timestamp answers “can you prove you did not change it yourselves”. They are different guarantees, at different costs, and vendors routinely blur them.

Most laboratories need the first and should understand that they are not getting the second unless they have arranged it. Capture at signing with a checksum defends against silent change by anyone without administrative access to the record store, which is the realistic threat in nearly every lab. A trusted timestamp defends against the lab itself, which matters in litigation, in patent disputes and in the rare regulatory case where the lab's own integrity is the question. Ask a vendor which one they mean when they say a record is tamper-evident. If the answer is a checksum, that is a good answer to the first question and no answer to the second, and a system that implies otherwise should be pressed on it.

What file evidence carries in this work

LabLynx has built electronic signature and document management capability into its systems for regulated laboratories for long enough to have watched the attachment problem surface in customer investigations, and the validation work around signatures is where the question of what a signature actually covers gets asked.

> **How LabLynx handles this**
>
> LabLynx One should let a draft record reference live files and, at signing, capture them into the record with a checksum, so the signed record holds its own evidence, says on its face when a file was too large to carry or a capture failed, and can report when the source has since changed. It does not provide third-party trusted timestamping unless that is arranged separately, and it will not imply that a checksum proves the lab did not alter its own records. If your need is that stronger guarantee, we will say so and help you arrange it. If it is stopping the file behind a signed result from quietly changing, that is the work we do.

Frequently asked questions

How do you prevent data loss in an ELN?

With backups, tested restores and retention that matches the regulation. But data loss is the loud, recoverable failure. The quiet one is a file behind a signed record changing without anyone noticing, and backups do not prevent it. Preventing that means capturing referenced files into the record at signing, with a checksum, so the record holds its own evidence.

What is ALCOA+ for attached files?

The same principles applied to the files behind a record: attributable, legible, contemporaneous, original, accurate, plus complete, consistent, enduring and available. A file that can be silently replaced after signing fails original, consistent and enduring. Capturing it into the record at signing, with a checksum, is how those principles are made true of the attachment as well as of the entry.

Does a checksum prove a file was not tampered with?

It proves the file has not changed since the checksum was taken. It does not prove the laboratory itself did not alter the file and re-compute the checksum, because the lab controls both. That stronger guarantee needs a trusted third-party timestamp. Most labs need the first and should know they are not getting the second unless they arranged it.

Should raw data files be attached to the record or linked?

Linked while the record is a draft, so a corrected instrument export is seen without re-attaching. Captured into the record with a checksum at the moment of signing, so the signed record holds its own evidence and stops depending on a folder somebody else controls. A file too large to capture should be linked, and the record should say so plainly.

What happens to a signed record if the source file is deleted?

If the record captured the file at signing, nothing: the record holds its own copy and can report that the source has gone. If the record only referenced the file, the signed record now points at nothing, the evidence behind an approved result is gone, and a backup may restore a version with no way to prove it is the one that was signed against.

Sources and references

1. 21 CFR 11.10 Controls for closed systems

   Electronic Code of Federal Regulations

2. 'GXP' Data Integrity Guidance and Definitions (2018)

   Medicines and Healthcare products Regulatory Agency

Brandon Holland

Marketing & Operations Digital Solutions Architect

Brandon Holland is LabLynx's Marketing & Operations Digital Solutions Architect. He builds and runs the digital systems behind the company's public surface, from the website and content architecture to the tooling that keeps product information accurate everywhere it appears. He writes about laboratory informatics from the systems side: how lab records get structured, searched, and kept defensible as a lab scales.

Read the current version online

https://www.lablynx.com/resources/articles/attachment-problem-signed-records/

Downloaded September 7, 2026. LabLynx builds laboratory informatics software: LIMS, ELN, LIS and lab automation.