Migration Risk and Mitigation

What can go wrong in a Microsoft 365 migration, and what stops it here

Tenrix Cloud Transformations

Every migration vendor says nothing will go wrong. That is not a sentence a risk committee can act on. This document sets out the specific ways a Microsoft 365 migration actually fails, and the exact mechanism in Tenrix's platform that prevents each one. Where a limitation is real, it is stated as plainly as the mitigation is.

A permission lands on the wrong person

The most common failure in a migration is not data loss. It is data arriving in the right place but granted to the wrong identity, because an automated match between two directories guessed, and nobody checked. Two people with the same name in different departments, an account that changed its username years ago, a contractor whose profile was never cleaned up: any of these can turn an automatic match into a real access problem.

What stops it here.
An automatic match is never treated as a decision. It is held as a suggestion until a named person on the migration confirms it. Nothing, mailbox, file, group membership or permission grant, moves for an identity until that confirmation exists, and the platform enforces this at the point of transfer itself, not by relying on a process someone could skip under pressure.

A report says “migrated” and something is quietly missing

A busy migration can look complete while individual items failed partway through, particularly under the throttling Microsoft applies to high-volume transfers. A count that only checks itself against its own tally, rather than against what the source actually held, will not notice the gap.

What stops it here.
Nothing is marked transferred until it is verified against the source, by byte count, item count or checksum depending on the content type. Verification is a precondition for that status, not a pass that runs afterward and gets skipped when time is short.

A recreated security policy fails open

Conditional Access and similar tenant-wide policies are rebuilt in the new tenant from the old one's configuration. If one of those comes up active before anyone has reviewed it, in particular before a break-glass account is confirmed excluded, it can lock every administrator out of a tenant with no other way back in.

What stops it here.
Every Conditional Access policy this platform creates in a target tenant is created disabled. There is no setting, flag or override that changes this. Enabling it is a deliberate action a person takes afterward, in the target tenant directly, once they have confirmed who is excluded.

The migration tool holds more access than the job needs

Being asked to grant write access to a tenant you have not agreed to migrate away from yet, or standing permissions wider than the task in front of them, is a normal thing for a vendor to ask and a reasonable thing to refuse.

What stops it here.
Reading the source tenant and writing to the target tenant are two separate applications, each with its own consent and its own declared scope. The application connected to a source estate cannot hold a write permission at all; that is a rule the platform checks before any call is made, not left to whoever configured the connection to remember.

The migration itself disrupts people still using the source

Moving a large estate quickly can mean calling Microsoft's APIs hard enough to get throttled, or worse, slowing things down for people still working in the source tenant while migration runs in the background.

What stops it here.
Every rate limit Microsoft signals is honoured exactly, never worked around with a shorter wait than instructed. Concurrency is adjusted continuously, per tenant and per kind of work, based on what the service is actually accepting right now, not a number fixed in advance and hoped for.

A multi-week migration loses its place

A tenant-to-tenant migration of any real size runs for weeks, not hours. Over that time, something will restart: a server reboots, a worker process dies, a network drops. A migration that keeps its progress only in memory loses it the moment that happens.

What stops it here.
Progress lives in one authoritative record, not in the memory of whichever process is running. A restarted worker reads that record and picks up exactly where the last one left off, rather than guessing or starting the estate over.

Nobody can say afterward what changed, or who changed it

A security review months after a migration, or an incident investigation, needs a plain answer: what was written into this tenant, when, and by whom. A log that can be edited or cleared does not answer that question, it only looks like it does.

What stops it here.
Every write into a customer tenant is recorded against the person who authorised it, with a timestamp, in a log that cannot be altered or deleted once written, not by a consultant and not by a platform administrator.

The final report overstates what actually moved

Some things do not survive a migration today. Certain chat history, content a customer encrypted with a key that does not exist in the new tenant, an old server's local permission groups: a report that stays silent about these is a compliance gap discovered by someone else, later, when it is expensive.

What stops it here.
A migration report never claims more than what is proven to have moved. Anything that did not, or could not, carry across is named explicitly in the same report as what succeeded, so the customer's own

decision about it is made before cutover, not found afterward.

What stays a human decision

None of the above is about removing judgement from a migration. It is about making sure the judgement that matters, confirming an identity, enabling a security policy, deciding what happens to content that cannot be carried over cleanly, is made by a named person who can be asked about it afterward, and never quietly defaulted by the software.