Five Stages, One Audit Trail: How Structured EFT Protects Treasury

Every treasury payment carries two kinds of risk. The first is the obvious one: the wrong amount going to the wrong place. The second is subtler and arguably more dangerous: the right payment going to the right place twice.

Both of these risks have a common root cause — process steps that are advisory rather than enforced, and the absence of a reliable trail showing what happened, when, and who approved it.

A structured EFT (Electronic Funds Transfer) workflow addresses both. Not by making payment operations more complicated, but by building each safeguard directly into the process so that it cannot be skipped, forgotten, or overridden without leaving a record.

Here is how each stage of the workflow reduces risk for a treasury team.

STAGE ONE: PAYMENT CONFIGURATION

Before a payment can move through any approval gate, someone must configure it— selecting the funding account, the payment channel, the beneficiary, and all the payment-type-specific data that the bank needs to process the instruction.

This stage exists because a payment generated by the settlements workflow contains the financial terms (amount, currency, value date, counterparty) but not the banking instruction. Those are different things. Conflating them — assuming that because a trade has been confirmed, the payment details are automatically correct — is one of the more common sources of failed payments in a treasury operation.

The configuration step forces someone to consciously set and verify the routing before the payment enters the approval queue. Mandatory fields are flagged by the system. Payment types define their own required data, and the system will not permit verification until all required fields are populated. The result is that a payment cannot accumulate approvals on the basis of incomplete or missing data.

STAGE TWO: VERIFICATION — THE CHECKER GATE

Verification is a distinct step, performed by a checker before approvals begin. It is the system's equivalent of "does this look right before we start getting sign-offs?"

The design here is deliberate. Verification is separated from approval for a reason: if a checker spots an error after approvals have already been recorded, the correct response is to go back to the beginning, not to patch the data and hope the approvers noticed. Unverifying a payment clears every existing approval automatically. This is not a punitive design choice — it reflects the logical reality that an approval given on the basis of incorrect data is not a valid approval.

The consequence is that verification is not a rubber-stamp step. A checker who unverifies a payment is doing the right thing, and the system supports that judgement by ensuring the full approval process restarts on clean data.

For a treasury team, this enforces two-person integrity at the data preparation stage, before any of the formal authorisation hierarchy becomes involved.

STAGE THREE: APPROVAL AGAINST THE BANKING MANDATE

The approval stage is where the Banking Mandate takes effect. Each payment is checked against the signing requirements for the funding account and amount tier, and approvals are recorded individually against each approver's user identity.

A colour-coded badge on the payment list makes the approval state immediately visible to anyone monitoring the queue:

No badge — no approvals recorded yet.
Orange — partially approved; the mandate requirement is not yet met.
Green — fully approved; eligible to be sent.

The value of real-time visibility here should not be understated. In a treasury team managing a large settlement run, the ability to see at a glance which payments are fully authorised and which are still waiting for a second signatory is operationally significant. It avoids the phone calls, emails, and spreadsheet trackers that teams without system-enforced workflows rely on — and which introduce their own error risk.

Only fully approved payments — green badge, all mandate requirements satisfied — can be included in a batch. The system blocks everything else at the point of

dispatch.

STAGE FOUR: SEND — A SINGLE CONTROLLED DISPATCH

When a user selects approved payments and clicks Send, the system does several things in a single atomic operation: it validates the selection, creates a batch, assigns a batch reference (PAYID), transmits the instruction to the bank, and stamps each payment with the batch reference as a permanent, non-editable link.

The significance of this being a single step is that it eliminates an entire class of error that exists in multi-step dispatch processes — the "generated but not sent" state, where a batch has been created but the transmission was not completed, and no one is sure whether the bank received the instruction.

Once the batch has been transmitted, each payment is locked. A locked payment cannot be included in a second batch. This is the first line of defence against double payment: the system structurally prevents a payment from being batched twice in its normal state.

The batch reference appears on each payment as a clickable link. Clicking it opens the batch detail screen, which shows every item in the batch and the bank's response — accepted, failed, and failure reason. This makes the reconciliation picture visible to the operations team as soon as the bank responds, without any manual data entry or file matching.

STAGE FIVE: REBATCH — CONTROLLED CORRECTION WITH EXPLICIT WARNINGS

Failed payments are an operational reality. A beneficiary account might be closed, a cut-off time might have been missed, or a formatting requirement for a specific payment type might have been overlooked. The question is not whether failures will happen but how the team responds when they do.

This is where the risk profile of a manual process diverges most sharply from a structured workflow. In a manual environment, a failed payment is corrected and resent by whoever handles the rejection. The double payment risk — the possibility that the bank's FAIL response does not mean the payment was not processed — is managed by institutional knowledge and, sometimes, by luck.

A structured rebatch workflow builds the double payment risk warning into the

process itself. When a failed payment is reopened for correction, the Integration Status panel shows the bank's original response in full, including the bank reference if one was assigned. When the user proceeds to approve the corrected payment, the system detects that this payment was previously sent and displays an explicit warning before approval is permitted.

The warning does two things. First, it forces the approver to confirm they have checked with the bank before proceeding — and to record that confirmation in writing as the override reason. Second, it creates an indelible audit record. The override reason, the user who entered it, and the timestamp are permanently attached to the payment's history.

The importance of the warning cannot be overstated. A FAIL status from the bank does not always mean the payment was not executed. Network timeouts, delayed acknowledgements, and cut-off mismatches can result in a payment that the bank has processed being returned with an error code. Rebatching in that state without first confirming the position with the bank is how double payments occur. The system makes this risk visible and creates a documented checkpoint — but it requires the user to take the decision consciously, with the facts in front of them.

THE AUDIT TRAIL AS A RISK MANAGEMENT ASSET

Every action in this workflow — configuration save, verify, unverify, approval, batch dispatch, bank response, status correction, duplicate override — is recorded with the user identity, date, and time. The full history is accessible on every payment via a single button.

For a treasury team, this serves several purposes beyond routine record-keeping. In the event of a query from the bank, a payment dispute, or an internal audit, the complete narrative of the payment is retrievable immediately: who configured it, who verified it, who approved it, when it was sent, what the bank said, and whether anyone overrode a duplicate warning and on what grounds.

This is not just a compliance requirement. It is the operational memory of your payments desk — the record that allows the team to answer questions accurately and quickly, and to identify patterns in failures that might point to a systematic problem in configuration or counterparty data.

A NOTE ON WHERE RISK ACTUALLY COMES FROM

Treasury payment risk rarely originates from malicious intent. It comes from well-meaning people working under time pressure, making reasonable assumptions that turn out to be wrong, or following a process that has gaps they were not aware of.

A structured EFT workflow does not assume bad faith. It assumes that people make mistakes, that data errors occur, that banks sometimes return confusing responses, and that deadline pressure can cause shortcuts that create downstream problems. It is built around the question: “If something goes wrong at this step, what prevents it from compounding into a more serious problem at the next step?”

The answer, at each gate, is the same: the system requires an explicit, recorded decision from an authorised person before the payment can proceed. No step can be silently bypassed. No error can be unknowingly carried forward into an approval or a dispatch.

For a treasury team, that consistency — enforced by the system rather than relying on individual discipline or memory — is the most significant risk reduction the framework provides.