Cross-Provider Identity Resolution for Enterprise Security Platforms

Fabrix

Abstract

Enterprise identity security requires a unified view of every person across all connected identity providers. Without this unification, the same human appears as multiple disconnected accounts, undermining access reviews, risk scoring, and compliance reporting.

We present a three-layer identity resolution system that combines deterministic correlation, heuristic scoring, and AI-assisted review to merge scattered user accounts into stable person entities. Deterministic rules handle the unambiguous majority — exact email matches and known IT conventions. Scored heuristics handle the structured-signal middle, cross-provider username and attribute matching. An LLM agent covers the contextual long tail, ambiguous cases where fixed-weight rules cannot reason about contradicting evidence, cultural name variation, or email semantics. Manual overrides cover the irreducible error rate, giving customers final authority. Stable identifiers ensure that downstream systems reference persons that do not change across provisioning cycles.

1. The Problem

Enterprise environments use multiple identity providers; Okta, Microsoft Entra ID, Google Workspace, GitHub, and others – each maintaining its own user directory with its own naming conventions, identifiers, and attribute schemas. A single employee might appear as:

- j.martinez@company.com in one identity provider
- JMartinez@company.com in another
- j-martinez_acme in a source-control platform with managed usernames
- jmartinez in an internal directory

Without resolution, the platform sees four separate identities. Access reviews miss permissions. Risk scores are fragmented. Compliance reports are incomplete.

The challenge is harder than it appears:

- **Names are not unique.** Two employees named "Sarah Chen" in the same organization may work in completely different departments, merging them would corrupt both their access records.
- **Email conventions vary across providers.** One system may use first.last@company.com, another flast@company.com, and a third may assign usernames with no email at all.
- **Employees have multiple accounts for different purposes.** Admin accounts, training accounts, staging accounts, and legacy accounts all legitimately belong to the same person but look like separate identities.
- **Corporate mergers introduce duplicate domains.** After an acquisition, the same person may have accounts at both company.com and acquired-company.com.
- **Terminated accounts retain old prefixes.** IT teams often rename deactivated accounts with prefixes like old_ or term_ rather than deleting them, creating additional identity fragments.

Any automated system must handle all of this while maintaining a hard constraint: never merge different people who happen to share a name or similar attributes.

2. Design Principles

- **Conservative by default.** A false merge, combining two different people into one, is far more damaging than a false non-merge. A missed correlation is visible and fixable; a wrong one silently corrupts downstream data. Every layer of the system is biased toward precision over recall.
- **Deterministic where possible, AI where necessary.** The vast majority of correlations are unambiguous: the same email appears across two providers, or an employee ID matches. These cases are handled by deterministic rules that run in seconds and produce repeatable results. The AI agent is reserved for the genuinely ambiguous cases, where multiple plausible interpretations exist and contextual judgment is required.
- **Stable identifiers.** Person entities must have identifiers that do not change when the underlying correlation logic evolves, when new providers are connected, or when accounts are added or removed. Downstream systems build indexes, caches, and audit trails on these identifiers. Instability breaks trust.
- **Human override as first-class citizen.** No automated system is perfect. Customers can force-merge or force-split persons at any time, and these decisions take precedence over all automated logic.

3. The Three Layers

LAYER 1: DETERMINISTIC CORRELATION

The first layer handles the straightforward cases through exact matching and well-understood transformations. These are correlations where the evidence is explicit and unambiguous, no inference is needed. This layer exists because these cases are the majority and can be resolved instantly with perfect precision.

Email matching is the strongest signal. When one user's primary email appears in another user's email aliases, or when emails match after stripping known old-account conventions (_old, _termed, term_ emp_ prefixes and suffixes), the system treats them as the same person. These patterns reflect real IT practices; terminated employee accounts are renamed with a suffix, rehired employees get a new account while the old one lingers – and they are unambiguous. For example, j.martinez@company.com and j.martinez_old@company.com are clearly the same person. Similarly, when a user's primary email appears as another user's alias (common after corporate email migrations), the match is definitive.

Connected components group transitive matches. If Account A matches Account B (via email alias) and Account B matches Account C (via normalized email), all three belong to the same person. The system uses iterative label propagation; each node adopts the minimum label among its neighbors, converging in a fixed number of iterations to efficiently compute these components without recursive queries.

This layer resolves the majority of correlations: typically 70–80% of person entities are fully resolved by email evidence alone.

LAYER 2: MULTI-ATTRIBUTE HEURISTIC SCORING

The second layer addresses cross-provider correlations where email evidence is absent or insufficient. A user in one system with username jmartinez and no email needs to be matched to j.martinez@company.com in another. Two accounts across providers share the same employee ID but have different email formats. This layer handles signals that are well-defined and can be reliably weighted, but it deliberately stops short of signals that require interpretation.

1. **Blocking** reduces the search space. Instead of comparing all N^2 pairs, the system generates candidate pairs that share at least one blocking key: same email domain, same department, same last-name prefix, username matching an email local part, same employee ID, and others. Twelve blocking strategies cover different correlation signals while keeping the candidate set manageable (even for organizations with thousands of users).
2. **Multi-attribute scoring** evaluates each candidate pair across 13 signals, each assigned a point value reflecting its confidence level:
   - Definitive signals (100 points): Employee ID exact match.
   - Strong signals (70 points): Full name + same department, full name + same manager, username equals email local part, username-to-fullname match, enterprise-managed username transformation matches email.
   - Medium signals (50–60 points): Exact name across different domains (for uncommon names), fuzzy name match (Jaro-Winkler ≥ 90%) with shared department or domain, normalized username match, email base pattern match (e.g., numbered accounts sharing the same base).
   - Supporting signals (30–40 points): Username matches name-derived pattern, first + last name components match.

   A pair correlates when either a single signal reaches 70 points (high confidence) or the total across all signals reaches 100 (multiple weaker signals accumulate). This dual-threshold design captures both definitive single-signal matches and cases where no one signal is strong enough alone but several converge.
3. **Preventing false merges.** The system uses multiple mechanisms to avoid incorrect correlations:
   - **Anti-correlation rules** block merges regardless of positive score. When two accounts have different employee IDs (both non-null), they are definitively different people, even if they share a name, department, and email domain. This is the strongest anti-correlation signal because employee IDs are assigned uniquely by HR systems.
   - **Threshold gating** means the vast majority of candidate pairs are discarded. Two users who share a department or last-name prefix but have no other signal never reach the 70/100 threshold. The blocking step is intentionally permissive (it generates candidates), but the scoring step is deliberately conservative (it rejects most of them).
   - **Minimum length requirements** prevent matches on trivially short strings; full names must be at least 5 characters for most scoring rules, and cross-domain name matches require at least 10 characters, avoiding false positives on common short names.
   - **Public domain exclusion** prevents the system from treating all Gmail or Yahoo users as potential matches simply because they share an email domain.

   Correlated pairs are fed into the same connected-components algorithm as email edges, producing unified person entities.

LAYER 3: AI-ASSISTED RESOLUTION

The first two layers are designed to be precise but incomplete. They handle correlations where the signals are unambiguous and can be expressed as deterministic rules or fixed-weight scoring. The third layer, an LLM agent, covers the interpretive gap: cases where the evidence is ambiguous or requires reasoning that cannot be reduced to fixed-weight rules.

Why heuristics are not enough

Heuristic scoring works by assigning fixed point values to specific signals and applying fixed thresholds. This design is reliable for well-defined patterns, but it has inherent limitations:

- **Heuristics cannot weigh contradicting evidence.** A fixed-weight system can detect that two persons share a name and a department (positive signals), but it cannot reason about the fact that their title, manager, location, and email domain all differ (contradicting signals). The heuristic system treats each signal independently; it cannot consider the overall picture and conclude that the volume of contradictions outweighs the matches.
- **Heuristics cannot interpret email semantics.** An email like user.bizdev@example.com may encode a name plus a role abbreviation. Heuristics can match exact patterns like first_initial + last_name, but they cannot reason about whether an email local-part plausibly maps to a specific person's name and job function, that requires linguistic understanding.
- **Heuristics cannot handle cultural name variation.** Determining whether two name variants represent the same person (a nickname, a married name, a transliteration from a non-Latin script) requires cultural and linguistic knowledge that cannot be captured in a fixed ruleset without constant expansion and maintenance.
- **Heuristics cannot distinguish person-unique values from defaults.** A phone number shared by two accounts is strong merge evidence unless that same number appears on hundreds of accounts, in which case it is an org default. Heuristics would need explicit allowlists; the AI reasons about frequency distributions dynamically.

What the AI adds

The AI agent's core advantage is **contextual, dynamic reasoning**. Unlike the heuristic layer's fixed point values, the agent evaluates each field's relevance based on the specific evidence in front of it. It is not given hard-coded weights for what constitutes a strong or weak signal. It classifies each field dynamically into two categories:

1. Person-unique fields (phone number, personal email, home address): matching values are strong merge evidence when genuinely personal, but meaningless when they appear across many accounts (org defaults).
2. Group-level fields (department, title, manager, location): correct values even when shared by many people. Not identity indicators by themselves, but powerful corroborating or contradicting evidence.

The agent applies this reasoning to all available fields, not a fixed checklist. For each field, it asks: "Is this value person-specific or a default? Does it corroborate or contradict the other evidence?" It then weighs the totality of evidence: how many group-level fields agree vs disagree, whether email patterns match naming conventions, whether name variants are plausible, to reach a judgment.

For split decisions, the agent examines a single person entity that may have been over-merged. It looks at the constituent user accounts, considers whether their email identities, organizational attributes, and naming patterns are consistent with a single person, and if not, recommends how to group the accounts into distinct people.
The agent also flags genuinely ambiguous cases, possible data-entry errors, field swaps, or mixed evidence that cannot be confidently resolved, for human review rather than making a potentially wrong automated decision.

Orchestration

The AI layer operates in three phases:

Phase 1: Global analysis

Before any LLM call, the system scans all person records across the tenant to build a global context:

- **Candidate group construction.** A union-find algorithm groups persons by name similarity (exact match, initial + last name, spelling variations) and email overlap (shared or similar email local parts, email-to-name convention matching). This produces candidate merge groups, clusters of persons that the heuristics believe might be the same human.
- **Split candidate identification.** Persons whose linked accounts have very different email identities (low similarity between email local-parts) are flagged as potential over-merges.
- **Org-wide email convention detection.** Across all persons, the system detects recurring email naming patterns; what fraction use first_initial + last_name, how common affix variants (admin, old, training) are, whether the same local-part appears across multiple domains (corporate mergers). These conventions become context for the LLM, so it can distinguish org-wide patterns from individual identity evidence.
- **Suppressed default value detection.** Fields that should be person-unique (like phone numbers) are scanned for values that appear across many unrelated accounts. These are identified as org defaults and will be explicitly suppressed in the LLM input, so the agent does not treat them as merge evidence.

Phase 2: LLM review.

For each candidate group (merge mode) or candidate person (split mode), the system calls the LLM with:

- The full person records – unique identifier, display name, email address, email aliases, and organizational attributes (department, job title, manager), plus all linked user accounts with their own attributes (email, login, department, title, location, manager, phone number, enabled status).
- The org-wide email conventions detected in Phase 1.
- The suppressed default values detected in Phase 1.
- The review mode (merge or split).

Large groups (> 60 persons) are subdivided by last name for manageable review windows. The LLM returns structured merge candidates, split candidates, and review-for-human candidates, each with an evidence-based reason.

Phase 3: Validation and persistence.

After each LLM call, the output is validated:

- Merge and split recommendations are checked against known person and account identifiers (rejects any hallucinated identifiers).
- Split recommendations must pass a strict gate: the person must genuinely have accounts with very different email identities.
- Previously reverted decisions are blocked from re-application (a customer who rejected an AI decision should not see it again).
- Customer-created merges and splits always take precedence over AI decisions.

Validated decisions are written to merge and split tables that the provisioning pipeline consumes on its next run.

Incremental evaluation.

The system computes a data fingerprint for each person record. On subsequent runs, only groups whose fingerprint has changed (new accounts, attribute updates, group composition changes) are re-evaluated. This keeps compute costs proportional to change volume rather than total person count.

Deployment and quality assurance

The AI layer is deployed with a phased supervision model designed to build confidence before granting autonomy.

- **Phased rollout:** In the initial phase, all AI merge and split decisions are reviewed internally before being applied, nothing touches production data without human sign-off. Once applied, every AI-driven change is tagged with its source (AI agent vs. manual), the model version, and an invocation identifier, so it is always distinguishable from rule-based or customer-initiated changes. As revert rates approach zero, the system graduates to applying decisions without pre-review. If revert rates are high, the prompt and model are iterated while the system remains in supervised mode.
- **Model selection:** The choice of LLM is driven by benchmarking across multiple models on real tenant data, evaluating both accuracy (false positive and false negative rates) and throughput. Through successive rounds of prompt refinement and heuristic pre-filter tuning, informed by false positive/negative analysis across all candidate models, we found that a smaller, faster model achieved zero false positives and zero false negatives relative to larger models, even on the most ambiguous merge candidates, while completing a full evaluation of thousands of persons in roughly half the time. The selected model is periodically cross-validated against stronger models on a benchmark set to catch regressions.
- **No blanket re-runs:** When the prompt or model is improved, the full pipeline is not re-run across all tenants. Prompt and model changes typically happen in response to revert feedback and those corrections are already applied by the revert itself. The improved logic produces better results on the next natural trigger. Selective re-evaluation is reserved for systematic issues affecting multiple tenants. Model and prompt versions are recorded alongside each decision for traceability, so it is always possible to identify which version produced a given recommendation.
- **Feedback loop:** Revert rates are tracked per tenant and per model version to measure real-world accuracy. Customers revert feedback into the understanding of edge cases and inform prompt refinements. Periodic benchmark comparisons against known-good decisions catch any regressions introduced by model or prompt changes.

4. How the Layers Work Together

The three layers plus manual overrides form an integrated system. Understanding how they interact is essential to understanding why the architecture works.

THE PROVISIONING CYCLE

On every provisioning run (typically daily), the deterministic and heuristic layers execute as part of the same pipeline:

1. **Ingest** – User accounts are pulled from all connected identity providers and normalized into a common schema.
2. **Correlate** – Layer 1 (email matching) and Layer 2 (heuristic scoring) build edges between accounts that should be merged. These edges, along with any previously written merge or split decisions (from the AI agent or manual overrides), are fed into the connected-components algorithm.
3. **Resolve** – The connected-components algorithm produces person entities. Each person gets a stable internal identifier (preserved from prior runs via priority-based matching) and a deterministic external identifier derived from it.
4. **Clean up** – Stale rows are removed: persons whose identities no longer exist, persons absorbed by a merge, orphaned split rows from reverted plans.

THE AI CYCLE

The AI agent runs on a separate schedule (typically after provisioning), operating on the person entities that provisioning produced:

1. **Scan** – The agent loads all current persons and their linked accounts. It runs heuristic pre-filters to identify candidate groups for merge and split review, detects org-wide conventions, and suppresses default values.
2. **Review** – For each candidate group that has changed since the last evaluation, the LLM reviews the full evidence and produces structured recommendations.
3. **Write** – Validated merge and split decisions are written to dedicated tables.

THE NEXT PROVISIONING RUN

On the next provisioning cycle, the system picks up the AI's decisions:

- **Merge records** cause the affected persons to be combined. When two or more persons are merged, the system designates one as the primary (the one with the most linked accounts) and absorbs the others. The primary person retains its stable identifier. The absorbed persons' identifiers are stored in metadata so they can be restored if the merge is later reversed.
- **Split records** cause new person entities to be created from the split plan, with the original person optionally kept or removed.

This means the AI agent's decisions do not take effect immediately. They are applied through the normal provisioning pipeline, which ensures all the same stability guarantees (identifier preservation, cleanup, transitive chain resolution) apply to AI-initiated changes just as they do to any other change.

MANUAL OVERRIDES

At any point, a customer can:

- Force-merge persons by creating a merge record. The next provisioning run applies the merge.
- **Force-split** persons by creating a split record. The next provisioning run creates the split.
- **Revert an AI decision** — the decision is deleted and a blocked key is recorded. The AI agent will not re-propose that specific merge or split.

Customer decisions always take precedence. If a customer has manually merged two persons, the AI agent will not propose splitting them. If a customer has manually split a person, the AI agent will not propose re-merging the constituent parts.

PRIORITY OF AUTHORITY

When multiple sources of truth conflict, the system applies a clear hierarchy:

1. **Customer manual decisions** – highest authority. Always applied.
2. **AI agent decisions** – applied unless overridden by a customer decision or a revert.
3. **Heuristic correlation** – the baseline. Produces the initial person entities that the AI and customer then refine.
4. **Deterministic email matching** – the foundation. Always runs and produces edges that feed into connected components.

This layered authority means the system improves over time: heuristics handle the bulk, the AI catches what heuristics miss, and customers correct what the AI gets wrong, with each correction feeding back into the system so the same mistake is not repeated.

5. Identifier Stability

Person entities carry a stable external identifier derived from an internal GUID. The GUID is a UUID assigned when the person is first created and preserved across all subsequent provisioning runs. The external identifier, being a deterministic hash of the GUID, is therefore stable.

Stability is maintained through a priority-based matching system during each incremental run:

1. **Exact identifier match** – the canonical identifier is unchanged.
2. **Split tracking** – when a person was split, the original GUID is stored in split metadata and restored if the split is reversed or the original identifier reappears.
3. **Merged GUID restoration** — when persons are merged, the absorbed persons' GUIDs are stored in the primary person's metadata. If the merge is later reversed, the original identifiers are restored.
4. **Identifier overlap** — when the canonical identifier changes due to new correlation evidence, the system finds the existing GUID by matching any of the person's constituent identifiers against existing records.

This design ensures that identifiers survive:

- New provider connections that add email evidence
- Correlation algorithm improvements that change the canonical identifier
- Merge and subsequent reversal
- Split and subsequent reversal

Identifiers change only on full table rebuild (rare, administrative operation) or when a person ceases to exist (all underlying identities removed).

6. Why This Architecture

Why not AI-only? LLM calls are slow, expensive, and non-deterministic. For the 80% of correlations that are unambiguous email matches, adding an LLM call would increase provisioning time from seconds to minutes with no accuracy benefit. Deterministic rules are also auditable a customer can see exactly why two accounts were merged (email alias match, employee ID match) without interpreting LLM reasoning.

Why not heuristics-only? Heuristic systems excel at well-defined patterns but fail on the long tail. They assign fixed weights to fixed signals, which means they cannot reason about how the totality of evidence should be interpreted. They cannot weigh contradicting group-level fields against identity signals, interpret email semantics, handle cultural name variation, or distinguish person-unique values from org defaults without explicit rules for every case. Attempting to encode all of these judgments as heuristic rules leads to an ever-growing ruleset with escalating maintenance cost and diminishing returns.

Why a conservative threshold? The dual-threshold design (single high-confidence signal ≥ 70 OR accumulated total ≥ 100) is deliberately conservative. In production, false merges combining two different people, cause cascading errors: wrong permissions in access reviews, inflated risk scores, compliance violations. False non-merges are visible (a customer sees duplicate person entries) and easily correctable via manual merge or AI-initiated merge. The system is biased toward the less-damaging error type.

Why stable identifiers? Downstream consumers (dashboards, audit logs, SIEM integrations, access review workflows) build indexes and references on person identifiers. If identifiers change on every provisioning run, these consumers must re-index, historical references become orphaned, and audit trails break. The GUID-based system with priority-based matching ensures that person identity is stable even as the underlying correlation logic evolves.

7. Conclusion

Identity resolution in enterprise security is a spectrum problem: most cases are unambiguous, some are pattern-matchable, a few require judgment, and a handful are genuinely uncertain. Applying a single approach across the entire spectrum wastes resources at one end and sacrifices quality at the other.

The three-layer architecture matches each resolution class to the appropriate method: deterministic rules for the unambiguous majority, scored heuristics for the structured-signal middle, AI for the contextual long tail, and manual override for the irreducible error cases. Stable identifiers and layered authority ensure that the system is both trustworthy and correctable.

The result is a person graph that accurately reflects the real humans behind the accounts, enabling identity security decisions, like access reviews, risk assessments, compliance audits, that operate on the right unit of analysis: people, not accounts.