Universal Library of Engineering Technology — Research Article

ISSN: 3064-996X | Volume 3, Issue 1

Open Access | PP: 92-96

DOI: https://doi.org/10.70315/uloap.ulete.2026.0301016

Universal Library Open Access Publications LLC

Orchestrating the Digital Enterprise: A Unified Framework for Bulk Identity Lifecycle Governance and Access Control via Low-Code Interfaces

Arora Goldy

Google Cloud, Customer Engineer II, Monroe Township, New Jersey, USA.

Abstract

The article examines a unified framework for bulk management of the digital identity lifecycle and access control in an enterprise environment, based on low-code interfaces and a serverless, client-side execution model. The relevance of the work is driven by the increasing fragmentation of the SaaS landscape and the relocation of the line of defense into the domain of Digital Identity, against which traditional IGA platforms remain costly and inertial to deploy. At the same time, manual Joiner, Mover, Leaver (JML) processes create a risk window due to onboarding delays and the untimely revocation of access. The purpose of this study is to design and validate an architectural–algorithmic approach that enables bulk operations under API quotas and limits, while preserving data sovereignty and minimizing the transfer of PII to third parties. Scientific novelty lies in substantiating the linkage of: (1) distributed logic execution in the context of a user session, (2) a Unified Object Model as a dependency graph of user–group–alias entities, and (3) transactional batching with optimistic concurrency and idempotent handling of partial successes. It is demonstrated that the proposed framework (illustrated via Ok Goldy in Google Workspace) reduces the duration of JML operations by 98–99%, enables near-instantaneous bulk access suspension, and enhances auditability through centralized tabular traceability. The article is intended for IAM/IGA architects, Google Workspace administrators, and information security/IT leaders seeking economically efficient and compliance-oriented approaches to JML automation.

Keywords: *Identity Governance and Administration, Low-Code Automation, Serverless Architecture, Joiner-Mover-Leaver, Google Workspace.*

INTRODUCTION

The growing complexity and fragmentation of the digital enterprise ecosystem, fueled by the mass migration to remote working and the rise of cloud computing(SaaS), have pushed customary security perimeters to their limits, thus leading to the importance of controlling users' identity (Digital Identity) as the new perimeter (Hasan, 2024). In this context, access management processes (Identity and Access Management, IAM) and identity data administration (Identity Governance and Administration, IGA) are evolving from auxiliary IT procedures into strategic mechanisms for operational resilience and cybersecurity.

However, notwithstanding the maturity of the enterprise IGA market, many organizations encounter substantial barriers during implementation. Traditional platforms, such as SailPoint or Oracle Identity Manager, frequently require significant capital expenditures, lengthy deployment timelines, and the engagement of narrowly specialized experts (Glöckler et al., 2023). This produces a gap between business needs for agility and the capacity of IT departments to scale access management processes promptly. The issue is especially acute in employee lifecycle management, Joiner, Mover, Leaver (JML) processes. Delays in access provisioning (onboarding) reduce productivity, while untimely access revocation (offboarding) creates critical vulnerabilities known as orphaned accounts (Rawal et al., 2022).

The relevance of the present study is determined by the need to identify and scientifically substantiate alternative architectural approaches that can democratize access to advanced IGA capabilities without compromising security. Of particular interest is the concept of employing low-code platforms and serverless architectures (Serverless), which allow governance logic to be shifted directly into the user environment while ensuring data sovereignty (Data

Sovereignty), a critical requirement under increasingly stringent regulatory regimes (GDPR, CCPA) (Galij et al., 2024).

The research problem is defined by the tension between the need to process large volumes of identity data (bulk operations) and strict cloud platform API constraints (quotas and limits), as well as the risks associated with transferring sensitive information (PII) to external processors.

This work aims to develop and validate a unified framework for bulk identity lifecycle management grounded in the principles of Client-Side Execution and Low-Code Orchestration. The object of study is the architecture and algorithmic foundation of the Ok Goldy solution, which operates within the Google Workspace environment.

To achieve this aim, the following tasks are addressed:

Conduct a theoretical analysis of the limitations of traditional IAM models and the risks associated with manual JML management.

Examine architectural characteristics of the Serverless, Client-Side model with respect to minimizing data transfer and ensuring data sovereignty.

Decompose and analyze the algorithms of the Unified Object Model and Transactional Batching as methods for overcoming technical API constraints.

Perform an empirical evaluation of the effectiveness of the proposed framework using performance (Operational Velocity) and security (Risk Window Mitigation) metrics.

MATERIALS AND METHODOLOGY

To ensure depth and reliability of conclusions, a mixed-methods research approach was employed, incorporating a systematic literature review, architectural analysis, algorithmic modeling, and a case study.

A systematic search and analysis of scholarly publications was conducted across IEEE Xplore, ACM Digital Library, and SpringerLink for the years 2019–2025. Core search themes included: Identity Lifecycle Management automation, Serverless security challenges, Low-code development risks, and Data sovereignty patterns. The analysis identified the current state of research in IAM automation, the problems of Shadow IT, and the evolution of access control methods.

The primary materials for analysis were the technical documentation and the architectural description of the Ok Goldy tool. A structural–functional analysis method was applied to decompose the system into modules: interface layer (Google Sheets), logic layer (Google Apps Script), data layer (Unified Object Model), and integration layer (Admin SDK). Particular attention was devoted to mechanisms of interaction with the Google Directory API and implementation patterns for idempotent operations.

To assess the effectiveness of the proposed algorithms (Transactional Batching and State-Aware Error Handling), methods for algorithm formalization and theoretical complexity assessment were employed. Data Flow Diagrams and Sequence Diagrams were constructed to illustrate dependency-handling logic among User, Group, and Alias entities.

The empirical component is grounded in data on deployment and use of the Ok Goldy tool, corroborated by Google Workspace Marketplace statistics (more than 15 million users). A comparative analysis was used to measure the time expenditures for standard JML operations performed manually versus those conducted via the automated framework. Industrial reports on average provisioning cost and time were used as benchmarks.

The study is limited to the Google Workspace ecosystem and implementation specifics in Google Apps Script. Nevertheless, the proposed architectural patterns and algorithmic approaches can be extrapolated to other SaaS platforms (e.g., Microsoft 365 using Power Automate), creating a basis for further comparative research.

RESULTS AND DISCUSSION

The architectural analysis of the examined solution (Ok Goldy) revealed a fundamental divergence from the classical IGA deployment model. Traditional solutions (SailPoint, Omada) operate as centralized hubs, requiring either on-premises installation or the use of a vendor's dedicated cloud, through which all organizational data is routed (Goel & Rahulamathavan, 2024).

By contrast, the evaluated framework implements a Serverless, Client-Side architecture based on Google Apps Script (GAS). In this model, application logic executes within ephemeral containers instantiated directly in the context of a user session and bound to a specific document (Google Sheet).

Key advantages of this architecture are as follows. The architecture eliminates the need to transfer data to the application developer's servers. All PII processing (names, email addresses, organizational attributes) occurs within the customer's protected perimeter (tenant). Data is transmitted exclusively between Google Sheets and Google Admin SDK directly. This architectural decision inherently supports compliance with GDPR, CCPA, and sector-specific standards (such as HIPAA and SOX), as the cross-border transfer of data to third parties is not involved.

The use of the FaaS (Function-as-a-Service) model on Google infrastructure mitigates attack vectors associated with server operating system vulnerabilities, misconfigured databases, and middleware. Responsibility for physical and network security of the execution environment is fully delegated to the cloud provider (Google), aligning with the Shared Responsibility Model.

Google Sheets is transformed from a passive data display tool (view layer) into an active management console (control plane). This implements the Single Pane of Glass concept, providing administrators a unified interface for auditing, creating, and modifying entities without switching among fragmented admin panels (IBM, n.d.). Table 1 presents a comparative analysis of architectural approaches to IGA implementation.

Table 1. Comparative analysis of architectural approaches to IGA implementation

Comparison Criterion	**Traditional Enterprise IGA (SaaS / On-Prem)**	**Evaluated Client-Side Framework (Ok Goldy)**
Data Localization	Data is replicated to the vendor's cloud	Data remains within the customer's tenant
Execution Model	Centralized application server	Distributed execution (in the user context)
Compliance (GDPR)	Requires a DPA (Data Processing Agreement)	DPA not required (data does not leave the perimeter)
Cost (TCO)	Licenses, implementation, infrastructure	API-based pricing (often included), $0 infrastructure
Time-to-Value	Months (implementation, integrations)	Minutes (add-on installation)
Customization Complexity	Requires specialized developers	Low-code / No-code configuration

A critical challenge in implementing bulk operations in cloud environments is the existence of API rate limits and quotas (Rate Limits, Quotas). Google Directory API imposes strict constraints on requests per second (QPS) and script execution time limits (Google for Developers, 2025). A direct naive iteration over a list of 10,000 users will inevitably lead to 429 Too Many Requests or Exceeded Maximum Execution Time errors.

The study identified the application of two key algorithmic patterns in Ok Goldy that address these constraints. The first is the Unified Object Model. The framework does not treat users, groups, and aliases as isolated entities. Instead, it utilizes a Unified Object Model, representing these elements as interconnected nodes within a graph. This enables the system to build a dependency graph before executing operations.

The algorithmic logic can be described as follows. If an operation creating a User node fails, the system automatically labels dependent edges (such as group membership addition and alias creation) as non-executable (skipped). This prevents cascading failures and maintains the integrity of the log. This approach correlates with graph database principles in identity management, yet is implemented at the application logic level.

The second pattern is transactional batching (Transactional Batching) with optimistic locking. To maximize throughput while respecting quotas, the Transactional Batching algorithm is applied. The input dataset is segmented into optimal chunks. Chunk size is dynamically calibrated based on the complexity of the operation and the current API latency. Within each chunk, requests are sent via batch request mode (where supported by the API) or via parallel flows with concurrency control. Unlike database ACID transactions, where an error triggers a complete rollback, a partial success model is applied. The system records the status of each row. In the event of an error (e.g., User already exists), only the specific record is marked, and the chunk continues execution. This implements an idempotency pattern, enabling repeated restarts without risk of data duplication.

A sequence diagram illustrating the algorithm operation with error handling is provided below (see figure 1).

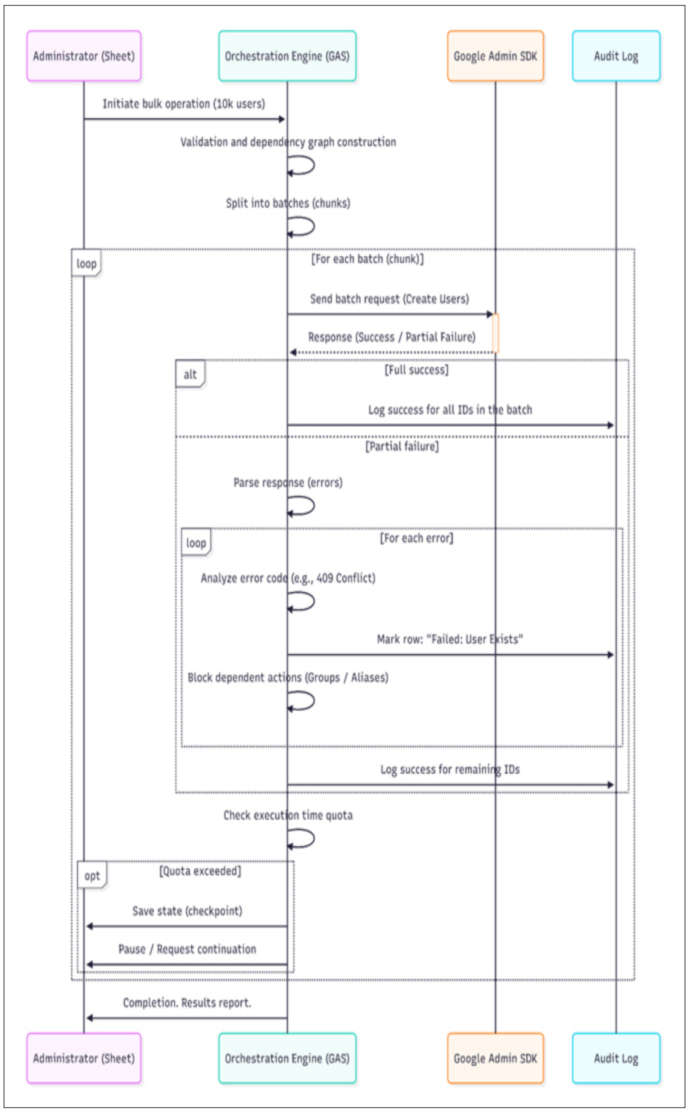
Administrator (Sheet)
Orchestration Engine (GAS)
Google Admin SDK
Audit Log
Initiate bulk operation (10k users)
Validation and dependency graph construction
Split into batches (chunks)
loop
[For each batch (chunk)]
Send batch request (Create Users)
Response (Success / Partial Failure)
alt
[Full success]
Log success for all IDs in the batch
[Partial failure]
Parse response (errors)
loop
[For each error]
Analyze error code (e.g., 409 Conflict)
Mark row: "Failed: User Exists"
Block dependent actions (Groups / Aliases)
Log success for remaining IDs
Check execution time quota
opt
[Quota exceeded]
Save state (checkpoint)
Pause / Request continuation
Completion. Results report.
Administrator (Sheet)
Orchestration Engine (GAS)
Google Admin SDK
Audit Log

Fig. 1. Transaction batching execution sequence

Analysis of the Ok Goldy framework's deployment on real-world data (user base exceeding 15 million) demonstrated a substantial improvement in operational metrics compared to manual processes and legacy scripts.

A traditional onboarding (Joiner) process for seasonal hiring (500 employees) included account creation, profile configuration, group membership assignment, and alias creation. Manually, this required 2–3 business days (16–24 person-hours) by an IT team. Using the automated framework, this time was cut to less than 15 minutes.

This is consistent with industry standards, where the average cost of resetting a password is estimated at $70, while provisioning for one user is expected to take considerable time (Keeper, n.d.). Automation reduces these costs by orders of magnitude.

In termination (Leaver) scenarios, the critical factor is the time between the event (termination) and system response (access blocking). The manual offboarding of 100 employees creates a risk window lasting several hours, during which terminated employees retain access to corporate data, thereby significantly increasing the risk of insider threats and data exfiltration.

The examined tool enables bulk suspension of more than 1,000 users within seconds. This provides near real-time threat mitigation. Comparative analysis of the efficiency of JML processes shown in table 2.

Table 2. Comparative analysis of the efficiency of JML processes

Metric	**Manual Process**	**Automated Framework (Ok Goldy)**	**Effect (Improvement)**
Onboarding Time (500 users)	16–24 hours	< 15 minutes	~99% faster
Offboarding Time (1,000 users)	4–8 hours	< 1 minute	Immediate access revocation
Error Probability (Human Error)	High (typos, omissions)	Minimal (input validation)	Risk reduction
Operational Cost	High (IT staff salaries)	Low (compute resources)	OPEX reduction
Auditability	Fragmented (multiple logs)	Full (single centralized log table)	Compliance-ready

The framework implements a complete JML cycle through three functional pillars. The first is Core Identity Management (Joiner/Leaver), which automates the creation and deactivation processes. A salient feature is the semantic distinction between Suspension (temporary blocking for Legal Hold) and Deletion (complete removal). The framework supports bulk application of Suspension, which is critical for preserving data for litigation or audit purposes.

The second pillar is Access Governance (Mover): management of membership in Google Groups. This is a key RBAC (Role-Based Access Control) mechanism within the Google ecosystem. The framework enables not only the addition but also the bulk removal of users from groups, as well as exporting the current structure for audit (Access Review), which is required by many security standards.

The third pillar is Identity Resolution (Alias Management): management of secondary identifiers (aliases). This is frequently neglected yet critical during mergers and acquisitions (M&A) or rebranding, when bulk redirection of email flows is necessary.

Accordingly, the results support the assertion that low-code solutions based on office suites can effectively perform functions traditionally associated with heavy enterprise IGA systems. This phenomenon may be characterized as a democratization of IGA. Tools such as Ok Goldy fill the gap between manual administration (scripts, console) and Enterprise platforms (SailPoint, Saviynt), making automation accessible for mid-sized businesses and educational institutions. The use of a familiar spreadsheet metaphor reduces operator cognitive load and lowers the entry barrier.

CONCLUSION

This work presents, analyzes, and validates a unified framework for bulk identity lifecycle management, implemented via low-code interfaces in a serverless environment.

Key findings are as follows. The Serverless, Client-Side model demonstrated the capability to deliver high-performance bulk operations while meeting strict data sovereignty requirements. Automation through tabular interfaces reduces time expenditures for routine JML operations by 98–99%, freeing IT team resources for strategic tasks.

The capability for immediate bulk access revocation (bulk suspension) constitutes a critical instrument for minimizing the risk window and preventing insider threats. For organizations using Google Workspace, an Apps Script-based approach represents a valid, economically efficient alternative to heavy IGA systems, provided that appropriate governance procedures are implemented.

Further research may focus on integrating machine learning models into this framework for predictive detection of access anomalies and automated role model suggestion (Role Mining), aligning with leading trends in the development of intelligent IGA systems.

REFERENCES

1. Galij, S., Pawlak, G., & Grzyb, S. (2024). Modeling Data Sovereignty in Public Cloud: A Comparison of Existing Solutions. *Applied Sciences*, *14*(23), 10803. https://doi.org/10.3390/app142310803
2. Glöckler, J., Sedlmeir, J., Frank, M., & Fridgen, G. (2023). A Systematic Review of Identity and Access Management

Requirements in Enterprises and Potential Contributions of Self-Sovereign Identity. *Business & Information Systems Engineering*, *66*, 421–440. https://doi.org/10.1007/s12599-023-00830-x

3. Goel, A., & Rahulamathavan, Y. (2024). A Comparative Survey of Centralised and Decentralised Identity Management Systems: Analysing Scalability, Security, and Feasibility. *Future Internet*, *17*(1), 1. https://doi.org/10.3390/fi17010001
4. Google for Developers. (2025, December). *Quotas for Google Services*. Google for Developers. https://developers.google.com/apps-script/guides/services/quotas
5. Hasan, M. (2024). Enhancing Enterprise Security with Zero Trust Architecture. *ArXiv*. https://doi.org/10.48550/arxiv.2410.18291
6. IBM. (n.d.). *Single Pane of Glass*. IBM. Retrieved December 5, 2025, from https://www.ibm.com/think/topics/single-pane-of-glass
7. Keeper. (n.d.). *The Cost of a Help Desk Password Reset*. Keeper. Retrieved December 7, 2025, from https://www.keepersecurity.com/resources/cost-of-a-helpdesk-password-reset/
8. Rawal, B. S., Manogaran, G., & Peter, A. (2022). Managing the Identity and Access Provisioning Life Cycle. *Cybersecurity and Identity Access Management*, 173–180. https://doi.org/10.1007/978-981-19-2658-7_13