Title

Context-Aware Hotkey Orchestration for Cloud Service Management Interfaces

Ideation

The present invention relates generally to computer-implemented systems, methods, and interfaces for cloud computing environments. More specifically, it relates to context-sensitive hotkey orchestration frameworks for Microsoft Azure and other multi-service cloud platforms, enabling faster, more secure, and less error-prone management operations across complex distributed infrastructures.

Background of the Ideation

Cloud platforms such as Microsoft Azure provide enterprises with a wide range of services-compute, storage, networking, security, identity, and DevOps. While powerful, these services typically require multi-click navigation through graphical user interfaces (GUIs) or manual command-line scripts.

Existing approaches have the following limitations:

1. Excessive Cognitive Load: Users must remember the location of deeply nested menus and service-specific commands.
2. Inefficiency: Common tasks, such as creating a resource group or assigning RBAC permissions, require multiple steps and context switching between portal tabs.
3. Error Prone: Mis-clicks or command typos may result in resource misconfigurations, downtime, or security exposure.
4. Lack of Personalization: Current automation tools (RPA, macros) do not adapt dynamically to user context (role, workspace, subscription).
5. Security Gaps: Generic scripts often bypass granular identity checks, resulting in administrative risk.

Thus, there is a need for a context-aware hotkey framework that maps single or chorded keystrokes to authorized Azure actions, dynamically resolving context (subscription, resource group, user privileges) while enforcing enterprise policies and security boundaries.

Summary of the Ideation

The invention provides a **hotkey-driven orchestration system** for cloud environments, comprising:

- A **hotkey capture module** that intercepts user key events from a client application (desktop, browser extension, CLI, or mobile app).
- A **context resolver** that identifies the current cloud workspace, subscription, role, and UI state.
- A **mapping engine** that translates the hotkey into an **authorized cloud operation**.
- An **execution handler** that invokes the corresponding **Azure REST API / SDK call** through authenticated channels (e.g., **MSAL OAuth tokens**).
- A **telemetry and audit subsystem** that records each action for compliance.
- **Error recovery routines** (retry, rollback, notification) to ensure reliability.

This system reduces task completion times, minimizes misconfiguration risks, improves accessibility, and ensures enterprise-grade governance.

Brief Description of the Drawings

Fig 1 High-level system architecture for the Azure Cloud Hot Keys framework.

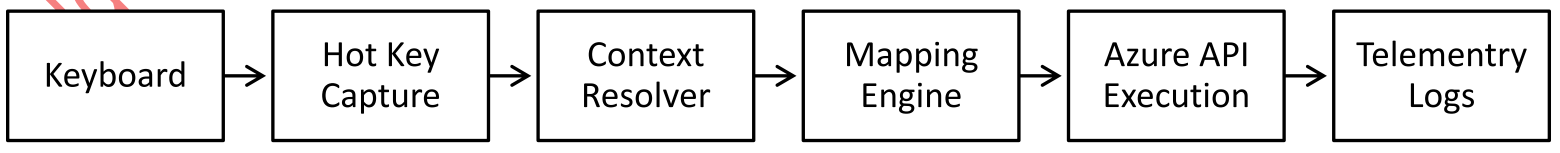


Fig 2 Hotkey decision flow (simplified)

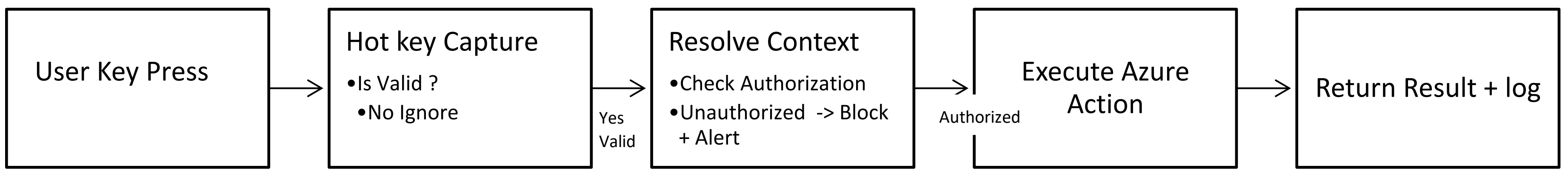


Fig 3: Example role-based context resolution diagram.

Ideation by Maulik Shah | https://maulikmshah.github.io/portfolio

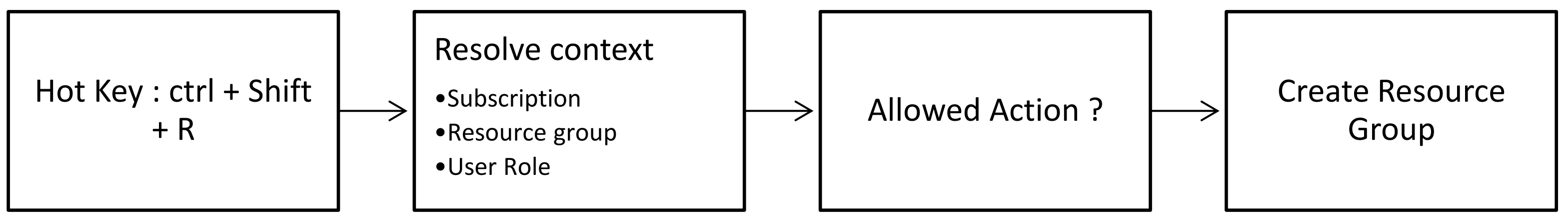
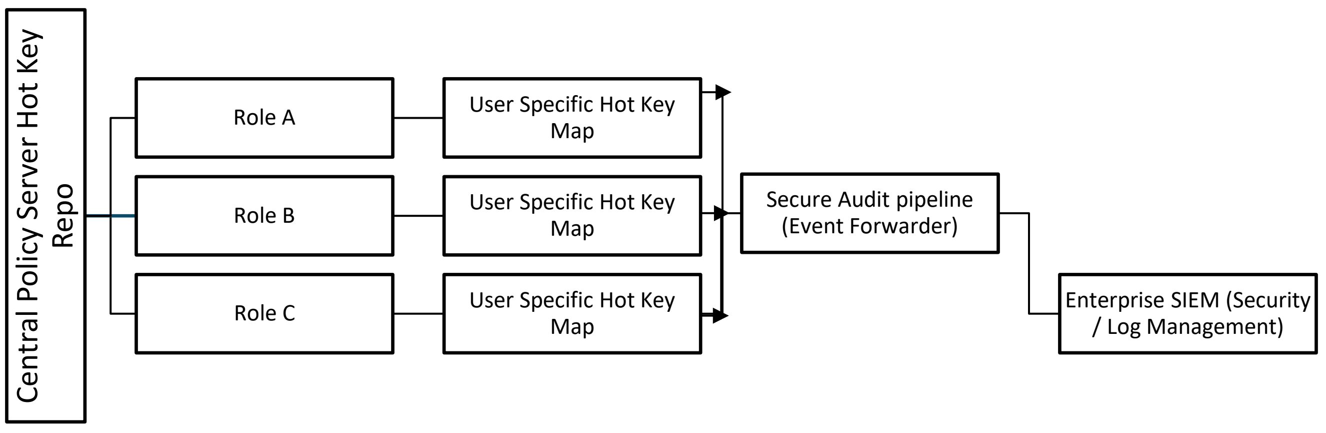


Fig 4 Multi-user enterprise deployment overview


Central Policy Server Hot Key Repo
Role A
Role B
Role C
User Specific Hot Key Map
User Specific Hot Key Map
User Specific Hot Key Map
Secure Audit pipeline (Event Forwarder)
Enterprise SIEM (Security / Log Management)

This figure shows:

- Central Policy Server distributing official hotkey maps.
- Roles (A, B, C) applying user-role-specific mappings and executing Azure actions.
- Secure Audit Pipeline forwarding execution records.
- SIEM (Security Information and Event Management system) consolidating logs for compliance/monitoring.

Detailed Description of the Invention

Overview

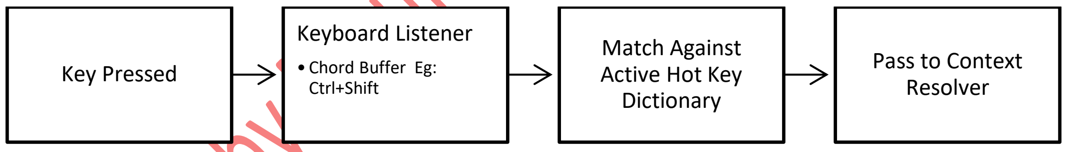
The invention provides a **context-aware hotkey orchestration framework** for cloud service platforms such as Microsoft Azure. Unlike static keyboard shortcuts or general-purpose macros, the disclosed invention dynamically **resolves context, verifies authorization, and invokes cloud-native APIs** while maintaining security, auditability, and enterprise governance.

Hotkey Capture Module

The Hotkey Capture Module intercepts keystrokes from the user's input device. It supports:

- Single-key mappings (e.g., F9 → Open Azure Dashboard).
- Chorded sequences (e.g., Ctrl+Shift+R → Create Resource Group).
- Global vs. Scoped mappings (e.g., global shortcuts across all Azure services vs. local context-specific ones).

Fig 5: Hotkey Capture Flow


Key Pressed
Keyboard Listener
• Chord Buffer Eg: Ctrl+Shift
Match Against Active Hot Key Dictionary
Pass to Context Resolver

Context Resolver Module

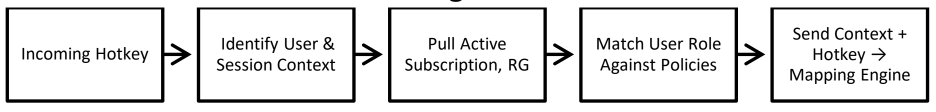
This module determines the execution environment at the time of hotkey invocation:

- Azure Subscription (e.g., DevOps-Canada-East).
- Resource Group (e.g., RG-Prod-Analytics).
- User Role (Owner, Contributor, Reader).
- UI State (Portal dashboard, CLI session, VS Code extension).

The resolver uses APIs such as:

- GET https://management.azure.com/subscriptions (for subscriptions).
- az account show (CLI).
- Cached session tokens (from MSAL).

Fig 6 Context Resolution


Incoming Hotkey
Identify User & Session Context
Pull Active Subscription, RG
Match User Role Against Policies
Send Context + Hotkey → Mapping Engine

Ideation by Maulik Shah | https://maulikmshah.github.io/Portfolio

Mapping Engine

The Mapping Engine translates hotkeys into Azure operations. It maintains a Hotkey Dictionary, stored either locally or distributed via a Central Policy Server (see FIG. 4).

Mapping rules may be:

Static: one hotkey always maps to the same operation.

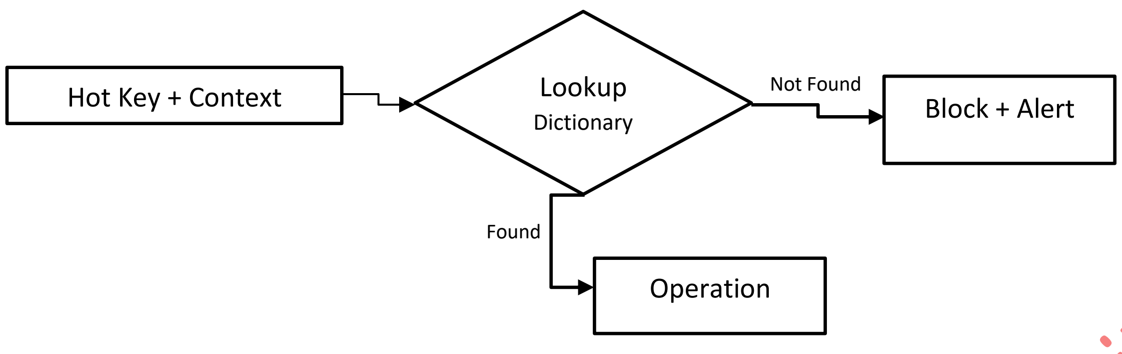
Dynamic: same hotkey maps differently based on role, context, or resource group.

Example:

Ctrl+Shift+R → *Contributor*: Create Resource Group.

Ctrl+Shift+R → *Reader*: Open Resource Group Dashboard (read-only).

Fig 7 Context Resolution

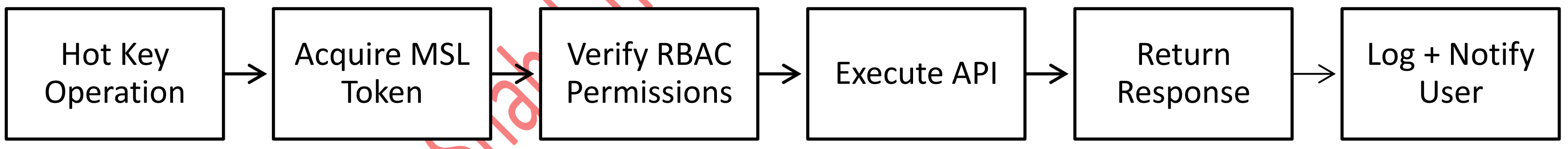


Execution Handler

The Execution Handler is responsible for securely invoking Azure APIs. It performs:

1. Token Retrieval: via MSAL (Microsoft Authentication Library).
   - Example: AcquireTokenInteractive for delegated permissions.
2. Authorization Check: Verifies role against RBAC policy.
3. Execution: Calls REST/Graph/CLI APIs.
4. Response Handling: Success/failure, retry logic, rollback if needed.

Fig 8: Execution Handler Pipeline

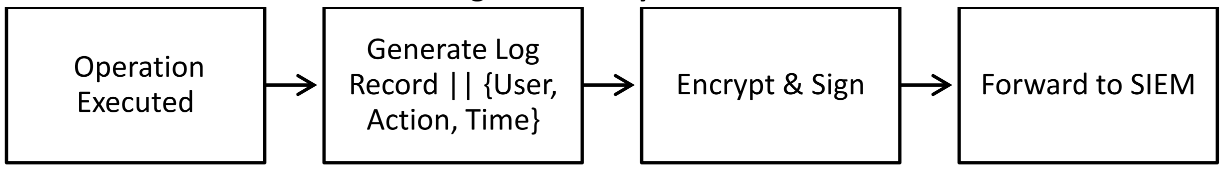


Telemetry & Audit Subsystem

The system maintains tamper-proof audit trails:

- Logs hotkey, context, action, timestamp, user ID.
- Encrypts logs before transmission.
- Forwards events to enterprise SIEM systems

Fig 9 Telemetry & Audit Flow



Error Handling & Recovery

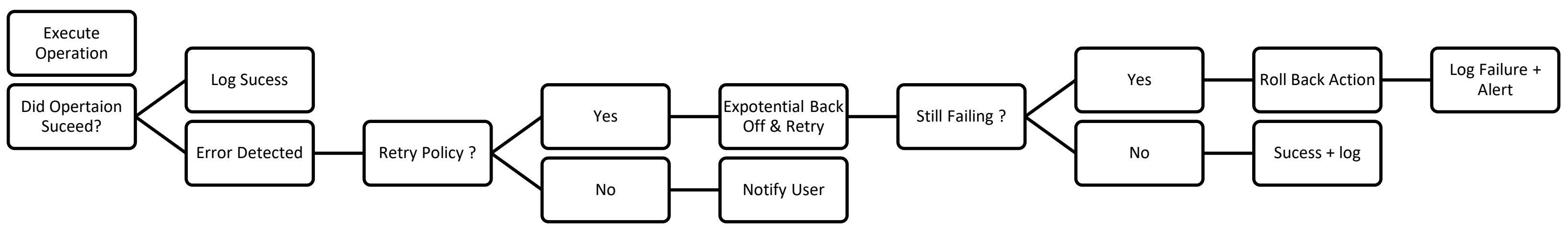
This subsystem ensures reliability in enterprise use cases where cloud operations may fail due to network issues, API rate limits, or insufficient permissions.

- Retry Logic: If an Azure API call fails due to transient errors (e.g., HTTP 429 Too Many Requests, 503 Service Unavailable), the handler automatically retries using exponential backoff.
- Rollback Mechanism: For multi-step operations, if one step fails after others succeed (e.g., VM created but network rule failed), the system triggers predefined rollback scripts to restore the prior state.

Ideation by Maulik Shah - https://maulikmshah.github.io/Portfolio

- User Notification: If retries and rollback do not resolve the issue, the user receives an immediate alert (desktop popup, email, or Teams notification).
- Logging: Both success and failure events are logged to ensure compliance and traceability

Fig10 Error Handling & Recovery Workflow

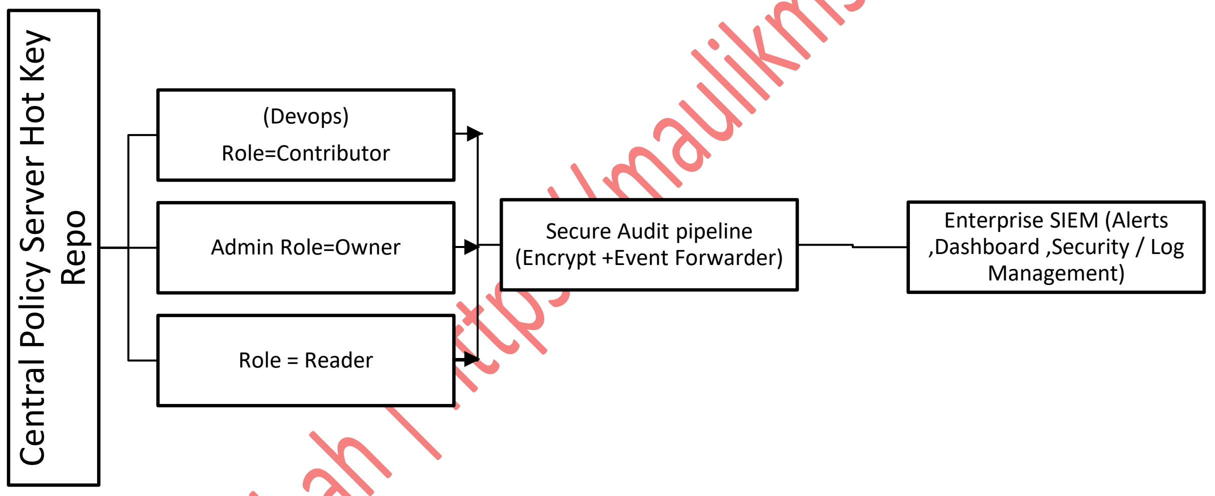


Multi-User Enterprise Deployment

In an enterprise setting, the invention scales to thousands of users with centralized governance.

- A Central Policy Server holds master hotkey dictionaries, mapped by role, department, or subscription.
- Each Client Agent (desktop, browser extension, CLI plugin) synchronizes with the server to fetch user-specific hotkey maps.
- A Secure Audit Pipeline encrypts execution logs and transmits them to the enterprise SIEM (Security Information and Event Management) system.
- This architecture ensures consistency, compliance, and role-specific enforcement across the organization.

Fig 11 Multi-User Enterprise Deployment (Expanded)


Central Policy Server Hot Key Repo
(Devops) Role=Contributor
Admin Role=Owner
Role = Reader
Secure Audit pipeline (Encrypt +Event Forwarder)
Enterprise SIEM (Alerts ,Dashboard ,Security / Log Management)

Ideation by Maulik Shah | https://maulikmshah.github.io/Portfolio

Alternative Embodiments

The invention is not limited to a desktop hotkey framework. Alternative embodiments include:

Browser Extension: Hotkeys trigger operations directly in the Azure Portal, intercepting DOM events and invoking APIs.

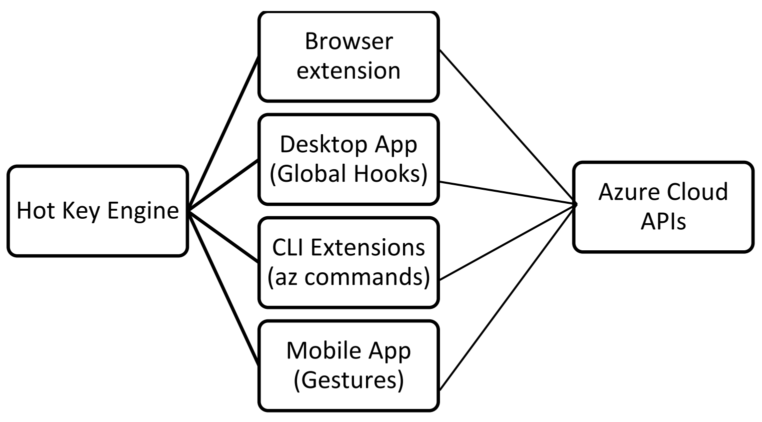
Desktop Application: A background agent captures OS-level hotkeys globally, enabling Azure operations even outside the portal.

CLI Extension: Hotkey-to-command mapping inside the az CLI, reducing typing and increasing speed for developers.

Mobile Application: Gesture-based equivalents of hotkeys (e.g., three-finger swipe = "deploy resource") for on-the-go cloud management.

Each embodiment shares the same core modules (capture -> context -> mapping -> execution ->telemetry) but adapts the input method and UI integration layer.

FIG. 12: Alternative Embodiments



Abstract

A system and method for executing context-aware hotkey operations in a cloud computing environment are disclosed. The idea provides a hotkey capture module for intercepting user inputs, a context resolver for determining subscription, resource group, user role, and user interface state, and a mapping engine for dynamically associating hotkey inputs with authorized cloud operations. An execution handler securely invokes cloud service APIs using authentication tokens while verifying authorization against role-based access control (RBAC) policies. A telemetry subsystem logs execution events, encrypts audit records, and forwards logs to an enterprise security information and event management (SIEM) platform. The invention improves operational efficiency, reduces misconfiguration risk, and enhances compliance by enabling secure, role-sensitive, and auditable hotkey-driven control of cloud resources.

Appendix A — Enabling Disclosure (Pseudocode)

This appendix provides representative pseudocode to further enable a skilled engineer to implement the invention without constraining it to a specific programming language or SDK. The same control flow applies to desktop agents, browser extensions, CLI plug-ins, and mobile embodiments described in the specification.

A1. Hotkey Capture & Chord Buffer

```
// Module: HotkeyCapture
onKeyDown(event):
  token = normalize(event)
  ChordBuffer.append(token)
  if isChordComplete(ChordBuffer):
    emit HotkeyEvent(chord=ChordBuffer.copy(), timestamp=now())
    ChordBuffer.clear()
```

A2. Context Resolver

```
// Module: ContextResolver
function resolveContext(hotkeyEvent):
  userId = Identity.currentUser()
  tokens = MSAL.getCachedTokens(userId)
  subscriptionId = Env.detectSubscription(tokens)
  resourceGroup  = Env.defaultRG
  role           = RBAC.effectiveRole(userId, subscriptionId, resourceGroup, tokens)
  return ExecutionContext{ userId, subscriptionId, resourceGroup, role, tokens }
```

A3. Execution Handler

```
// Module: ExecutionHandler
function execute(op, ctx):
  if not RBAC.authorize(ctx.tokens, ctx.role, op.requiredPermissions):
    return fail("Unauthorized")
  request = Azure.buildRequest(op, ctx)
  resp = Azure.call(request, token=ctx.tokens.accessToken)
  if resp.success:
    return success(resp.payload)
  else:
    return fail(resp.error)
```

Ideation by Maulik Shah | https://maulikmshah.github.io/portfolio