# Red Hat Ansible Automation Platform (AAP) Governance & Configuration Document

## 1. Introduction

### 1.1 Purpose

* Define AAP governance, organizations, role-based access control (RBAC), and permissions.
* Ensure secure, scalable, and standardized automation practices for {{ ORGANIZATION\_NAME }}.

### 1.2 Scope

* Covers user access, job execution, inventory management, and compliance policies.
* Excludes custom playbook development (handled by individual teams).

### 1.3 Audience

* **Platform Engineers & Administrators**: Maintain platform governance.
* **Engineers & Operators**: Execute and develop automation.
* **Security & Compliance Teams**: Audit platform usage.

## 2. AAP Architecture & Execution Model

### 2.1 High-Level Architecture

* AAP consists of Control Nodes (Orchestration) and Execution Nodes (Playbook Execution).
* Execution Environments (EEs) run in user space on Execution Nodes for security.

### 2.2 Control Plane vs. Execution Plane

* **Control Nodes**: AAP Web UI, API, RBAC Management, Job Scheduling.
* **Execution Nodes**: Run jobs inside isolated containers, fetch EEs from {{ AAP\_REGISTRY\_URL }}.

### 2.3 Security & Isolation

* RBAC controls all job executions and inventory modifications.
* Only Execution Nodes run playbooks, preventing privilege escalation risks.
* Centralized inventory and credential management to avoid unauthorized system modifications.

## 3. AAP Organizations & Role-Based Access Model

### 3.1 Organization Structure

* **Prod**: Hosts approved automation for production environments.
* **Develop**: Used for testing and validating playbooks before production deployment.

#### Why This Model?

* **Simplicity & Scalability**: Avoids redundant team-based orgs.
* **Code & Automation Parity**: Matches Dev → Prod branching strategy.
* **Transparency**: All roles have read access to both organizations.

### 3.2 Role-Based Access Control (RBAC) Model

* No individual user permissions → Access granted only via team roles.
* **RBAC Provider**: {{ AAP\_RBAC\_PROVIDER }}
* **Identity Management System**: {{ IDENTITY\_MANAGEMENT\_SYSTEM }}
* Platform Engineers govern inventories & job templates, preventing license overuse.

### 3.3 Team Role Definitions

|  |  |
| --- | --- |
| Role | Responsibilities |
| **Platform Administrator** | Full control over AAP resources. |
| **Platform Architecture** | Admin in Develop, Read-only in Prod. |
| **Platform Engineer** | Develops & maintains automation workflows. |
| **Platform Operator** | Executes approved automation workflows. |
| **Platform Auditor** | Read-only access for compliance monitoring. |

### 3.4 Permissions Model Overview

* **Inventories**: Centrally managed by Platform Admins, preventing license overuse.
* **Job Templates**: Created by Platform Engineers; Operators can execute but not modify.
* **Credentials**: Stored in {{ AAP\_CREDENTIAL\_STORAGE }} with RBAC restrictions.

## 4. Platform Configuration

### 4.1 Load Balancer Integration

* Load Balancer: {{ AAP\_LOAD\_BALANCER }}
* Health checks & failover policies.

### 4.2 Authentication & Access Control

* LDAP/Okta Authentication integration.
* RBAC enforcement via {{ IDENTITY\_MANAGEMENT\_SYSTEM }}.
* SailPoint integration for user lifecycle management.

### 4.3 Execution Environment Configuration

* Standardized Execution Environment (EE) image lifecycle.
* Secure EE hosting in {{ AAP\_REGISTRY\_URL }}.
* Building & Updating EEs using Ansible Builder.

### 4.4 Collections Management

* Approved collections whitelist & versioning policies.
* Architectural review process for new collections.
* Automating collection updates via {{ AAP\_REGISTRY\_URL }}.

### 4.5 Host Inventory & Groups

* Centralized inventory management by Platform Engineers.
* RBAC-driven inventory access control.
* Preventing overutilization of licensed hosts in {{ AAP\_INVENTORY\_MANAGEMENT }}.

## 5. Team Onboarding & Role Integration

* AAP organization structure and its rationale.
* Role-based access control (RBAC) policies and team roles.
* How permissions are granted and policy against individual user permissions.

### 5.1 AAP Team Roles & Permissions

* **No Individual Permissions**: Access is assigned only through team roles mapped to {{ AAP\_RBAC\_PROVIDER }}.
* **SailPoint Integration**: Team membership is managed through SailPoint to ensure identity governance.
* **Platform Engineers Manage Resource Assignments**: Assign inventories, job templates, and projects to team roles, but group membership is handled via {{ IDENTITY\_MANAGEMENT\_SYSTEM }}.
* **Transparent Access**: All roles have read access to both organizations to enable cross-team collaboration.

## 6. Security & Compliance

### 6.1 Hardening Guidelines

* User-space only container execution on Execution Nodes.
* RBAC & network segmentation for security isolation.

### 6.2 Sensitive Data Management

* All API keys & credentials stored in {{ AAP\_CREDENTIAL\_STORAGE }}.
* No hardcoded secrets in playbooks.
* Platform Admins manage all secrets.

### 6.3 Compliance Monitoring & Auditing

* Compliance audits conducted {{ AAP\_COMPLIANCE\_AUDIT\_FREQUENCY }}.
* Weekly security log reviews in Splunk.
* Automated compliance reporting via Ansible.

## 7. Enhancements & Continuous Improvement

### 7.1 Roadmap for Enhancements

* New integrations: ServiceNow, SIEM, Cloud APIs.
* Automating governance policies (e.g., Role assignment automation).

### 7.2 Retrospective Processes

* Quarterly usage & access reviews.
* Team feedback sessions & automation improvements.

## 8. Appendices

### 8.1 Glossary

* AAP, RBAC, EEs, Quay, SIEM, SailPoint, etc.

### 8.2 Reference Links

* Red Hat AAP Documentation
* CIS Benchmark for RHEL Security
* Ansible Best Practices Guide