ASTRAflow Platform: Architecture Design Document

1. Introduction

1.1 Purpose and Scope

This Architecture Design Document outlines the architectural blueprint for **ASTRAflow**, a next-generation business process orchestration and automation platform designed for cloud-native deployment. ASTRAflow is engineered to provide sophisticated workflow automation while adhering to strict requirements for data residency, security, and compliance through a decoupled, hybrid deployment model.

The core objective of ASTRAflow is to enable organizations to seamlessly design, deploy, monitor, and manage complex workflows in AWS cloud environments with minimal operational overhead. This document details the structure, interactions, and responsibilities of the platform's distinct architectural components, ensuring secure, scalable, and maintainable operations across multi-tenant deployments.

Document Scope

This document covers:

- Platform architecture overview and design principles
- Component responsibilities and interactions
- Data flow and security architecture
- Multi-tenancy implementation approach
- Deployment models and operational considerations
- Technology stack and integration patterns

1.2 The Split-Plane Architecture

ASTRAflow is built upon a **Split-Plane Architecture**, strategically dividing functionality into two complementary operational planes:

- **Control Plane (SaaS)**: Managed, proprietary services operated by the ASTRAflow platform team
- **Execution Plane (BYOC)**: Customer-hosted, secure execution components deployed within client AWS accounts

This architectural separation ensures that:

1. **Critical execution** of workflows and **access to client-specific resources** remains strictly within the client's environment (Bring-Your-Own-Cloud - BYOC)
2. **Operational burden** of design, deployment, and management is handled by ASTRAflow as a managed Software-as-a-Service (SaaS) offering
3. **Data residency requirements** are honored with sensitive execution logs and audit journals remaining in customer accounts

4. **Security compliance** (GDPR, HIPAA, SOC 2) is simplified through clear separation of concerns

Architectural Principles

ASTRAflow adheres to enterprise-grade architectural principles:

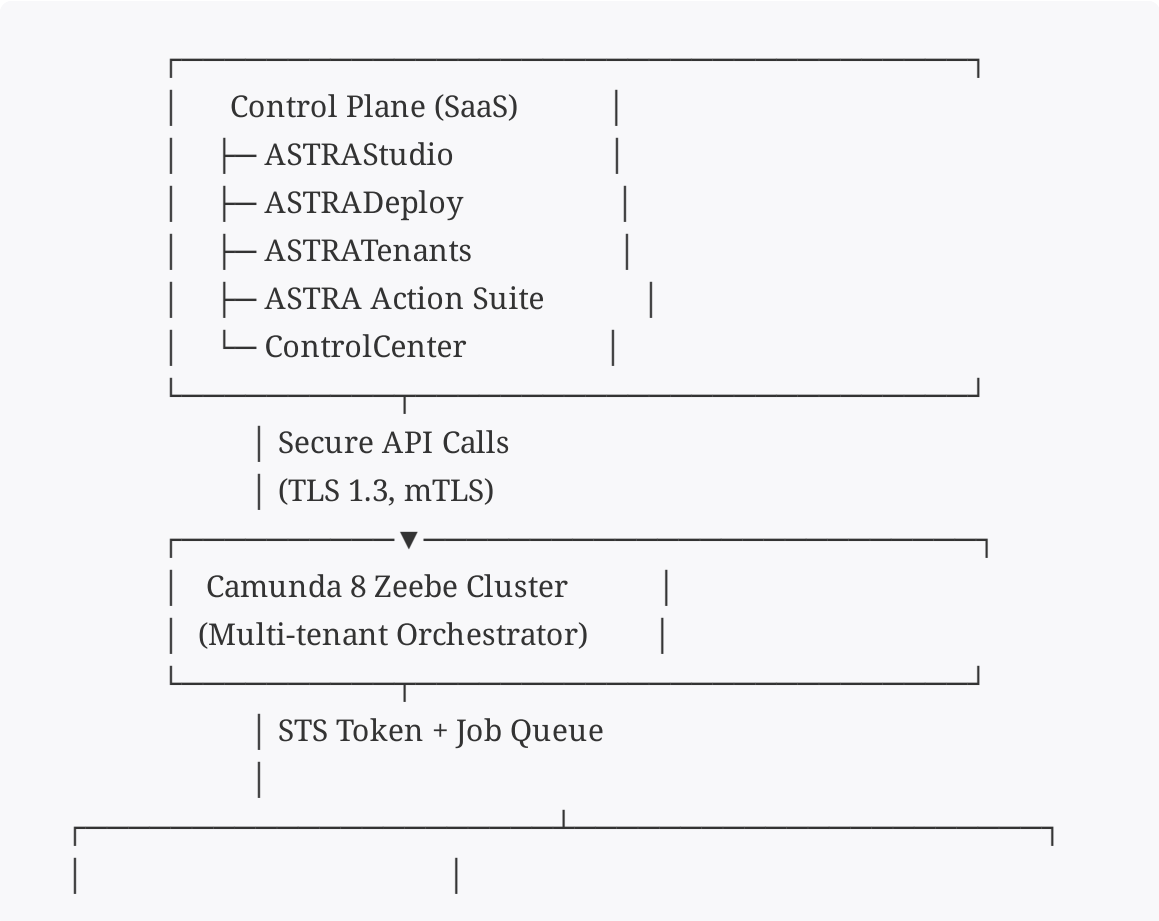
- **Separation of Concerns**: Design and execution responsibilities are logically and physically separated
- **Zero-Trust Security**: All communications are authenticated, encrypted, and authorized
- **Data Residency**: Sensitive data remains in customer-controlled environments
- **High Availability**: Resilient components enable graceful degradation and fault tolerance
- **Multi-Tenancy**: Secure resource isolation enables efficient resource utilization
- **Auditability**: All actions are logged with immutable audit trails for compliance

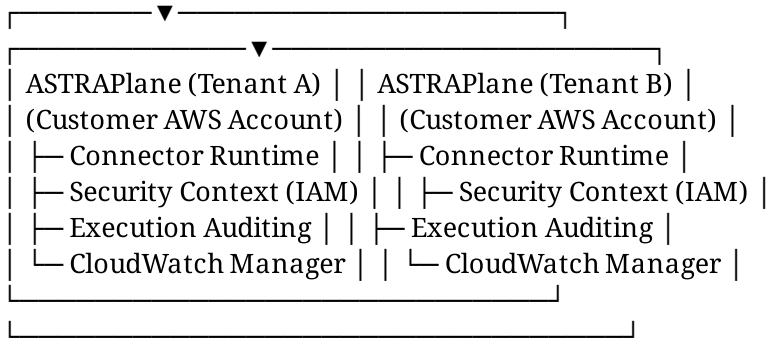
---

2. Architecture Overview

The ASTRAflow platform is composed of three primary, logically separated components:

- Control Plane (SaaS) - Hosted and managed by ASTRAflow
- ASTRAPlane (Execution Plane - BYOC) - Deployed in customer AWS accounts
- Camunda 8 Zeebe Cluster (Workflow Orchestrator) - Shared multi-tenant engine





---

3. Control Plane (SaaS)

The Control Plane is the fully managed, multi-tenant SaaS component hosted and operated by the ASTRAflow platform team. It serves as the central workflow management and governance module, providing a single pane of glass for design, deployment, and operational oversight.

3.1 ASTRAStudio: Workflow Design and Monitoring

Purpose: Provides the graphical interface for designing, testing, and monitoring workflows; offers real-time execution visibility.

Key Capabilities

- **Visual Workflow Designer**: Drag-and-drop interface for designing complex workflows for alert resolutions, business continuity planning (BCP) events, and other automation requirements
- **Workflow Testing**: Test workflows with sample data and validate connectivity before deployment to production
- **Execution Monitoring**: Real-time dashboard displaying active workflow instances, execution progress, and performance metrics
- **Audit Trail Examination**: Detailed audit logs for each workflow execution with timestamped events and decision paths
- **Instant Deployment**: One-click deployment capability with immediate effect in the Zeebe cluster
- **Tenant Isolation**: ASTRAStudio isolates onboarded workflows and execution details based on tenant configuration, ensuring multi-tenant security
- **Execution Plan Visualization**: Visual representation of workflow execution paths, decision points, and job assignments
- **Error Diagnostics**: Comprehensive error messages and debugging tools for workflow troubleshooting

  #### Technical Implementation

  - **Frontend**: React-based single-page application with real-time WebSocket connections
  - **State Management**: Tenant-aware state isolation using tenant context propagation
  - **Data Access**: Filtered query results based on tenant authorization
  - **Integration**: REST API calls to ControlCenter and ASTRATenants for tenant-aware operations

3.2 ASTRADeploy: Deployment Lifecycle Management

Purpose: Manages the promotion and deployment lifecycle of workflows to the underlying Camunda 8 Zeebe engine.

Key Capabilities

- **Deployment Orchestration**: Manages workflow promotion through development, staging, and production environments
- **Version Control**: Maintains deployment history with ability to rollback to previous versions
- **Deployment Progress Tracking**: Real-time status updates during workflow deployment
- **Workflow Details Registry**: Comprehensive metadata for each deployed workflow including version, deployment timestamp, and tenant association
- **Tenant-scoped Deployment**: All deployments are scoped to specific tenants, preventing cross-tenant workflow execution
- **Deployment Validation**: Pre-deployment checks ensure workflow definitions conform to BPMN 2.0 specifications and are compatible with available connectors
- **Concurrent Deployments**: Supports simultaneous deployments across multiple tenants without conflicts
- **Rollback Capability**: Automatic or manual rollback to previous workflow versions in case of deployment failures

Deployment Flow

1. User selects workflow version in ASTRAStudio
2. Workflow definition is transmitted to ASTRADeploy
3. ASTRADeploy validates BPMN structure and connector availability
4. Deployment is created in Zeebe with tenant identifier
5. Deployment confirmation is returned to ASTRAStudio
6. Monitoring begins for new workflow instances

3.3 ASTRATenants: Tenant Lifecycle and Access Management

Purpose : Manages the entire client lifecycle, including provisioning, configuration, and user/access management for each isolated tenant.

Tenant Model

In ASTRAflow, a tenant represents a distinct business unit or application scope that is associated with specific workflows, users, and AWS environments. The tenant model provides the foundational organizational structure for the entire platform.

Tenant Characteristics

- **Distinct Identity: Each tenant has a unique identifier used to tag all workflows, executions, and audit data**
- **Resource Association: Tenants are associated with:**
  - **Specific workflows and their versions**
  - **AWS service configurations and VPC resources**

        * IAM roles and cross-account access policies
        * User and group access permissions
        * Execution audits and logs
    - Multi-Tenant Assignment: Users can be associated with multiple tenants; workflows can be linked to one or more tenants
    - AWS Configuration: Each tenant maintains its own AWS service configuration that governs access within its VPC or across multiple VPCs

Key Capabilities

- Tenant Provisioning: Automated setup of new tenants with configuration templates
- User Management: Addition, removal, and role assignment of users to tenants
- Access Control: Definition and enforcement of permissions (owner, editor, viewer, auditor)
- AWS Configuration Management: Storage and management of AWS account IDs, VPC configurations, and IAM role ARNs
- Quota Management: Definition and enforcement of resource limits per tenant (workflow count, execution rate, storage)
- Configuration Audit: Complete audit trail of all tenant configuration changes
- Self-Service Onboarding: APIs for customer-initiated tenant provisioning and configuration

Tenant Isolation Strategy

- Logical Isolation: Tenant IDs are embedded in all database queries and Zeebe operations
- Access Control: Row-level security ensures users can only access tenant-specific data
- Network Isolation: ASTRAPlane components in separate AWS accounts provide network-level isolation
- Cryptographic Isolation: Cross-tenant data is cryptographically separated where applicable
- Audit Isolation: Execution audits are stored in customer accounts, preventing cross-tenant access

3.4 ASTRA Action Suite: Connector and Action Management

Purpose : Manages all ASTRA Actions and Zeebe connectors, orchestrates container image creation, and provides action templates to ASTRAStudio.

Key Capabilities

- Action Definition: Create, manage, and version custom actions that integrate with external systems
- Connector Management: Wrapper layer over Zeebe connectors providing:
    * Authentication credential management
    * Rate limiting and circuit breaker patterns
    * Retry policies and error handling
    * Request/response transformation
- Container Orchestration: Automated building and pushing of connector container images to AWS Elastic Container Registry (ECR)
- Image Versioning: Semantic versioning of connector images with rollback capability

- Action Catalog: Searchable catalog of available actions and connectors for workflow designers
- Custom Action Development: SDKs and frameworks for developing custom actions
- Action Documentation: Auto-generated documentation for each action including parameters, outputs, and error scenarios

Connector Image Pipeline

1. Action definition is created in ASTRA Action Suite
2. Dockerfile and configuration are generated based on action type
3. Container image is built using multi-stage build for optimized size
4. Image is pushed to ECR with tenant-specific tags and access controls
5. ASTRAPlane pulls image during deployment using IAM-authenticated access
6. Image is stored in local ECR within customer account for execution

3.5 ControlCenter: Execution Orchestration and Authorization

Purpose : The decision-making engine that determines workflow execution eligibility, performs pre-validation, and provisions secure credentials for resource access.

Key Capabilities

- Alert-to-Workflow Mapping: Rules engine that maps incoming alerts to appropriate workflows based on alert content, source, and tenant configuration
- Execution Eligibility: Pre-flight checks before workflow execution including:
  - Workflow availability and deployment status
  - Tenant subscription status and feature entitlements
  - Resource quota availability
  - Time-based scheduling constraints
- Credential Provisioning: Generation of AWS STS tokens for secure cross-account resource access
- Tenant Configuration Lookup: Retrieves and applies tenant-specific configuration for workflow execution context
- Job Queue Management: Manages job assignment to appropriate ASTRAPlane workers based on tenant affinity
- Execution Context Creation: Constructs complete execution context including:
  - Tenant identifier
  - AWS STS credentials with minimal privileges
  - Connector configuration and secrets
  - Execution environment variables
- Audit Event Generation: Records all execution decisions for compliance and debugging

AWS STS Token Generation

ControlCenter follows AWS security best practices for temporary credential provisioning:

- Short-Lived Tokens: Generated with 1-hour expiration (configurable minimum)

- **Principle of Least Privilege: IAM policies grant only required permissions for specific workflow execution**
- **MFA Enforcement: Optional MFA requirement for sensitive operations (configurable per tenant)**
- **Session Tagging: Tags session with tenant ID, workflow ID, and execution ID for audit trail**
- **Regional Endpoints: Uses AWS regional STS endpoints for improved latency and reliability**
- **Token Validation: Validates credentials before passing to ASTRAPlane**

4. ASTRAPlane: Execution Plane (BYOC)

The ASTRAPlane is the client-specific component, designed to be deployed and hosted within the client's own AWS account. This component acts as a highly secure bridge between the managed Control Plane and the client's internal resources.

4.1 Architecture Overview

ASTRAPlane operates as a decoupled, autonomous system that:

- **Requires no direct database access from the Control Plane**
- **Communicates with Zeebe cluster through job polling mechanism**
- **Executes workflows using customer-provided AWS credentials**
- **Maintains full audit trails within customer accounts**
- **Operates independently if Control Plane becomes unavailable**

4.2 Security Context

Purpose : Manages client-side access, including IAM Roles and credential handling for secure access to other AWS services during workflow execution.

Key Responsibilities

- **IAM Role Management: Creation and maintenance of least-privileged IAM roles for connector execution**
- **Credential Rotation: Automatic rotation of temporary credentials with configurable intervals**
- **Secret Management: Integration with AWS Secrets Manager for storing sensitive credentials (API keys, database passwords, etc.)**
- **Cross-Account Access: Setup of trust relationships for accessing resources in other AWS accounts**
- **Policy Validation: Enforcement of customer-defined security policies and access constraints**
- **Credential Caching: Secure, in-memory caching of valid credentials with automatic refresh**
- **Compliance Verification: Validation that IAM policies comply with organizational standards**

IAM Role Design

Each ASTRAPlane instance operates with a principal IAM role that:

1. Has trust relationship with Zeebe cluster

2. Allows assuming tenant-specific roles
3. Has minimal permissions (only ability to assume downstream roles)
4. Is scoped to specific AWS resources

Tenant-specific roles are assumed by connectors and grant:

- EC2 access for infrastructure automation
- S3 access for data processing and storage
- RDS access for database operations
- Lambda invocation for serverless operations
- SNS/SQS for messaging and event handling
- Systems Manager for parameter and secret retrieval

4.3 Connector Runtime

Purpose : Hosts the secure, tenant-specific connector container images required to execute service tasks and integrate with client endpoints.

Architecture

The Connector Runtime operates as:

- Container Orchestrator: Runs connector containers using ECS (recommended) or EKS
- Job Worker: Polls Zeebe cluster for jobs tagged with tenant identifier
- Credential Provider: Injects AWS STS credentials and secrets into connector execution context
- Result Handler: Completes jobs in Zeebe with workflow variables or fails jobs on error
- Error Recovery: Implements exponential backoff and circuit breaker patterns for failed connections

Deployment Models

Option A: AWS ECS (Recommended)

- Fargate launch type for serverless container execution
- Service definition with auto-scaling based on job queue depth
- CloudWatch for container monitoring and logging
- IAM task roles for secure credential injection

Option B: AWS EKS

- Kubernetes cluster for advanced workload management
- Horizontal Pod Autoscaling (HPA) based on Zeebe job queue metrics
- Namespace isolation per tenant
- Service mesh (Istio) for advanced networking and security

Container Image Management

- Images are pulled from ECR repositories scoped per tenant
- Image tags follow semantic versioning (e.g., connector-http:2.1.0)
- Image scanning enabled for vulnerability detection
- Images are stored with private access controls
- Registry mirroring for disaster recovery

Job Processing Flow

1. Connector Runtime polls Zeebe with tenant filter
2. Zeebe assigns job to worker with:
- Job ID and task ID

- Workflow variables
- Connector configuration
3. Worker retrieves AWS STS credentials from ControlCenter
4. Worker launches connector container with:
- AWS_ACCESS_KEY_ID / AWS_SECRET_ACCESS_KEY / AWS_SESSION_TOKEN
- CONNECTOR_CONFIG with endpoint and authentication
- WORKFLOW_VARIABLES as environment variables
5. Connector executes task (e.g., call AWS API, invoke Lambda, query database)
6. Connector returns results
7. Worker completes job in Zeebe with output variables

4.4 Execution Auditing

Purpose : Responsible for storing sensitive execution and audit journals, ensuring that critical compliance data remains resident within the client's dedicated cloud environment.

Audit Data Collection

All execution events are captured including:

- **Process Events: Process instance started, completed, failed, terminated**
- **Task Events: Task activated, completed, failed, retried**
- **Variable Events: Variable created, updated, deleted with values**
- **Error Events: Errors thrown, errors caught, escalations**
- **User Events: Manual task approvals, form submissions**
- **Connector Events: Connector invocations, success/failure, performance metrics**
- **Decision Events: Decision table evaluations, outcomes**

Storage Architecture

Audit data is stored in customer-controlled storage:

- **S3: Primary storage for long-term audit trails with versioning enabled**
- **CloudWatch Logs: Real-time streaming for operational monitoring**
- **ElasticSearch: Optional full-text search capability for large-scale audit analysis**
- **DynamoDB: Optional for real-time query access to recent executions**

Audit Data Schema

Each audit record includes:

```
{
"tenantId": "tenant-uuid",
"executionId": "execution-uuid",
"workflowId": "workflow-id",
"timestamp": "ISO-8601 timestamp",
"eventType": "TASK_COMPLETED | PROCESS_STARTED | ...",
"taskId": "optional-task-id",
"taskName": "optional-task-name",
"executionContext": {
"userId": "optional-user-id",
"sourceAlert": "optional-source-identifier",
```

"duration": "milliseconds",
"status": "SUCCESS | FAILURE | RETRY"
},
"variables": { /* sanitized workflow variables */ },
"auditHash": "SHA-256 hash for tampering detection"
}

Compliance Features

- Immutable Audit Trail: Audit records are write-once, read-many using S3 Object Lock
- Encryption: All audit data encrypted at rest and in transit
- Access Control: Strict IAM policies limiting audit access to authorized users only
- Retention Policies: Configurable retention periods aligned with regulatory requirements
- Integrity Verification: Cryptographic hashing of audit records for tampering detection
- Export Capability: Ability to export audit trails in standardized formats (JSON, XML)

4.5 CloudWatch Observability Access Manager

Purpose : Monitors ASTRAPlane components and provides operational visibility for customer support and internal debugging.

Observability Components

- CloudWatch Metrics: Custom metrics for connector performance, job processing rates, error rates
- CloudWatch Logs: Structured logging from all ASTRAPlane components
- CloudWatch Dashboards: Pre-built dashboards for monitoring health and performance
- CloudWatch Alarms: Automatic alerting on anomalies (high error rates, timeout increases, etc.)
- X-Ray Tracing: Distributed tracing for end-to-end visibility of workflow execution
- Performance Profiling: CPU, memory, and network utilization tracking

Key Metrics

- Job processing latency (p50, p95, p99)
- Job success/failure rates per connector type
- Container resource utilization (CPU, memory, network)
- Queue depth and processing throughput
- Connector API response times and error rates
- AWS credential refresh latency
- Audit trail ingestion rate

Access Management

- Customer Access: Limited to their tenant's logs and metrics only
- Shared Dashboards: Pre-configured dashboards automatically created per tenant
- Log Retention: Configurable retention (default 30 days, customizable per compliance needs)
- Cross-Account Access: Authorized ASTRAflow operations team can assume customer role for support

- Audit of Access: All access to CloudWatch data is logged for compliance

---

5. Camunda 8 Zeebe Cluster: Workflow Orchestrator

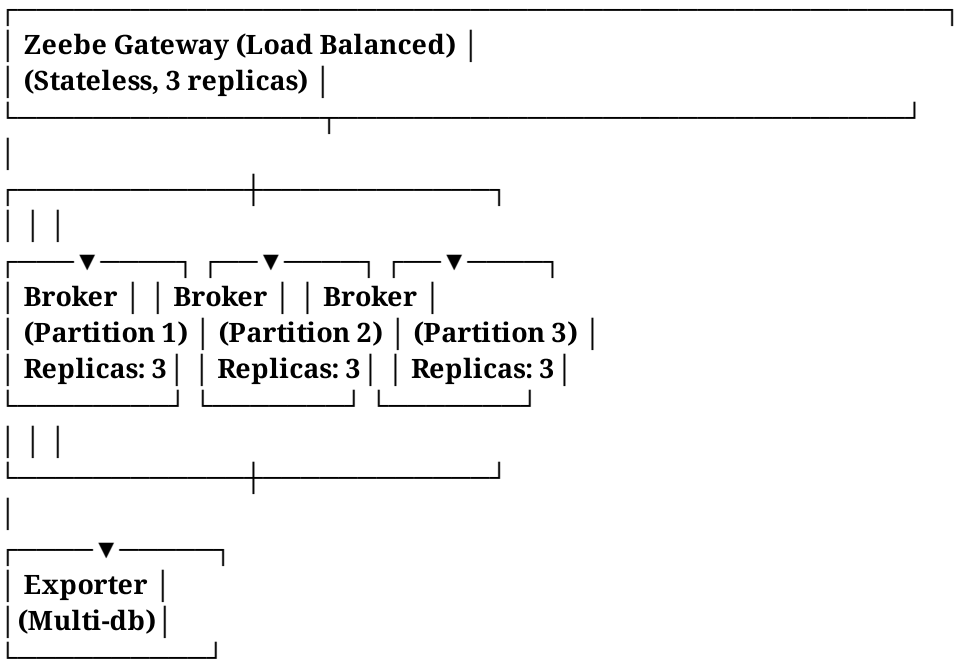
The Camunda 8 Zeebe Cluster is the high-performance, resilient, and scalable engine responsible for orchestrating and executing the automated workflows.

5.1 Zeebe Architecture

Core Components

- Zeebe Brokers: Distributed workflow engines that execute process definitions and manage process instances
- Zeebe Gateway: Stateless entry point that forwards requests to brokers
- Zeebe Exporter: Extracts events from Zeebe's event log for external consumption (audit trails, analytics)
- Identity: Authentication and authorization service managing tenant-based access control

High Availability Configuration



Deployment in AWS

Zeebe cluster is deployed using:

- Kubernetes (EKS) for container orchestration
- Helm Charts for templated deployments
- StatefulSets for Zeebe brokers (maintaining stable identity and storage)
- Deployments for gateway and exporter components
- PersistentVolumes for broker state (using EBS volumes)
- Network Policies for zero-trust networking

5.2 Multi-Tenancy Implementation

Zeebe implements multi-tenancy using Logical Isolation approach:

Tenant Identifier Propagation

All operations include a tenant identifier that:
- Is provided by the client application or extracted from JWT token
- Is used to tag all process definitions, instances, jobs, and variables
- Is enforced at the broker level to prevent cross-tenant data access
- Is validated in Identity before any operation is processed

Isolation Mechanism
- Process Definitions: Scoped to tenant, preventing other tenants from deploying or modifying
- Process Instances: Isolated by tenant ID in index, only visible to authorized users
- Jobs: Tagged with tenant ID, ensuring job workers only receive jobs for their tenant
- Variables: Stored with tenant ID, preventing cross-tenant variable leakage
- Audit Events: Exported with tenant ID for tenant-specific audit trails

Query Enforcement

All queries against Zeebe include implicit tenant filters:

```
SELECT process_instances
WHERE process_id = 'alert-response'
AND tenant_id = 'customer-abc' /* Implicit filter */
```

5.3 Scalability Characteristics

Horizontal Scaling
- Partitioning: Multiple partitions enable parallel workflow processing across different keys
- Broker Replicas: Each partition has 3 replicas for fault tolerance
- Elastic Gateway: Gateway replicas scale based on request throughput
- Job Worker Pool: ASTRAPlane connector runtimes scale based on job queue depth

Performance Targets
- Throughput: 100,000+ process instances per second
- Latency: < 500ms end-to-end workflow execution (task to completion)
- Availability: 99.99% uptime with multi-region failover
- Data Retention: Multi-year retention with tiered storage

5.4 Integration with Control Plane
- Deployment: ASTRADeploy sends BPMN definitions to Zeebe via gRPC API
- Monitoring: Zeebe Exporter sends events to Control Plane for ASTRAStudio monitoring
- Authorization: Identity component integrates with ASTRATenants for user access control
- Job Publishing: ControlCenter publishes jobs to Zeebe for ASTRAPlane execution
- Results: Zeebe publishes job completion to exporter for audit trail storage

---

6. Data Flow and Communication

6.1 Workflow Deployment Flow

1. Designer creates workflow in ASTRAStudio
│
▼
2. Workflow definition sent to ASTRADeploy
│
▼
3. Validation checks:
- BPMN structure verification
- Connector availability check
- Variable schema validation
│
▼
4. Tenant configuration lookup (ASTRATenants)
│
▼
5. Deployment to Zeebe with tenant ID tag
│
▼
6. Deployment confirmation to ASTRAStudio
│
▼
7. Monitoring enabled for new instances

6.2 Workflow Execution Flow

1. Alert or trigger received by ControlCenter
│
▼
2. ControlCenter determines workflow to execute
│
▼
3. Pre-validation:
- Tenant status check
- Resource quota verification
- Feature entitlement validation
│
▼
4. AWS STS token generation for tenant account
│
▼
5. Workflow instance created in Zeebe with tenant ID
│
▼
6. ControlCenter publishes job to Zeebe
│
▼
7. ASTRAPlane Connector Runtime polls for jobs

|
▼
8. Job assigned to connector worker
|
▼
9. Worker executes task:
- Retrieves STS credentials
- Pulls connector image from ECR
- Injects credentials and variables
- Launches container
- Connector invokes AWS service or endpoint
|
▼
10. Results returned to Zeebe
|
▼
11. Audit event stored in customer S3
|
▼
12. Next task activated or workflow completed

6.3 Communication Security

Control Plane to Zeebe
- **Protocol: gRPC over TLS 1.3**
- **Mutual Authentication: mTLS using client certificates**
- **Encryption: All data encrypted in transit**
- **Network: VPC-internal communication only (no public internet)**

Zeebe to ASTRAPlane
- **Protocol: gRPC over TLS 1.3 (job worker connections)**
- **Authentication: Zeebe service account with tenant restrictions**
- **Polling-based: ASTRAPlane initiates outbound connections only**
- **No Ingress: ASTRAPlane receives no inbound connections from internet**

Control Plane to ASTRAPlane
- **Protocol: HTTPS REST API (optional, for monitoring purposes)**
- **Authentication: Signed AWS SigV4 requests with IAM roles**
- **Firewall Rules: Restricted to ASTRAflow's IP ranges**
- **PrivateLink: Optional AWS PrivateLink for private connectivity**

7. Multi-Tenancy Model

7.1 Tenant Isolation Strategies

1. Data Isolation
- **Tenant ID is embedded in all database queries**
- **Row-level security policies enforce tenant boundaries**
- **Indexes are built with tenant ID as leading column for performance**
- **Cross-tenant data access is cryptographically impossible**

2. Compute Isolation
- **Each tenant's ASTRAPlane runs in separate AWS account**

- Connector containers tagged with tenant ID
- Job workers process jobs only for assigned tenant
- Execution audits stored in tenant-specific S3 buckets

3. Network Isolation

- Zeebe clusters separated by tenant (optional for ultra-sensitive workloads)
- VPC peering restricted to specific ASTRAPlane instances
- Security groups enforce least-privilege ingress/egress
- CloudFront + WAF protects Control Plane from tenant-based attacks

4. Cryptographic Isolation

- Audit data encrypted with tenant-specific keys (optional)
- Key management via AWS KMS with per-tenant key policies
- Encryption at rest and in transit
- Encrypted backups with tenant-specific decryption keys

7.2 Tenant Onboarding Process

Phase 1: Provisioning

- Customer initiates tenant creation via ASTRATenants API
- Tenant record created with unique ID
- AWS account ID and region specified
- VPC configuration provided

Phase 2: Configuration

- IAM roles created for cross-account access
- ECR repository created for tenant connectors
- S3 buckets created for audit trails
- CloudWatch log groups created

Phase 3: Deployment

- ASTRAPlane infrastructure deployed to customer account
- Connectivity validated between Control Plane and ASTRAPlane
- Zeebe Exporter configured for tenant-specific event stream
- Monitoring and alerting enabled

Phase 4: Activation

- Tenant marked as active
- Workflows can begin deploying
- Users assigned to tenant
- Execution metrics baseline established

---

8. Security Architecture

8.1 Authentication and Authorization

Control Plane Authentication

- User Authentication: OAuth 2.0 / OpenID Connect integration
- API Authentication: JWT tokens with tenant claims
- Service Authentication: AWS IAM roles for service-to-service communication
- MFA: Optional enforcement for sensitive operations

Authorization Model

- Role-Based Access Control (RBAC): Predefined roles (owner, editor, viewer, auditor)

- Attribute-Based Access Control (ABAC): Fine-grained policies based on tenant, resource, and action
- Delegation: Tenant administrators can delegate roles to other users

Zeebe Authorization

- Identity Integration: All Zeebe operations validated against Identity service
- Tenant Membership Verification: User's tenant list checked before accessing resources
- Job Worker Authorization: Connector runtime authenticated via certificate-based mTLS

8.2 Encryption

- Encryption in Transit: TLS 1.3 for all communications, mTLS for service-to-service
- Encryption at Rest: AWS KMS encryption for databases, S3 buckets, and encrypted snapshots
- Key Management: Customer-managed keys (CMK) option for sensitive environments
- Secret Management: AWS Secrets Manager for connector credentials and API keys

8.3 Audit and Compliance

- CloudTrail: AWS API calls logged for compliance auditing
- Execution Audits: Immutable audit trail of all workflow executions
- Access Logs: All user actions in ASTRAStudio logged with user ID and timestamp
- Compliance Reporting: Automated reports for SOC 2, GDPR, HIPAA requirements
- Data Residency: Sensitive data remains in customer-selected region

8.4 Threat Mitigation

- Rate Limiting: API rate limits prevent brute force and DoS attacks
- Input Validation: All inputs validated and sanitized before processing
- SQL Injection Prevention: Parameterized queries used throughout
- CSRF Protection: CSRF tokens on all state-changing operations
- Security Headers: HSTS, CSP, X-Frame-Options enforced on Control Plane
- Dependency Scanning: Regular scanning for vulnerable dependencies
- Penetration Testing: Quarterly security assessments

---

9. Operational Considerations

9.1 Deployment Architecture

Control Plane Deployment

- Platform: AWS ECS on Fargate or EKS
- Multi-Region: Active-active deployment across regions for disaster recovery
- Database: Multi-master PostgreSQL with read replicas
- Caching: Redis for session management and rate limiting
- CDN: CloudFront for static assets and API acceleration
- Load Balancing: Application Load Balancer with health checks

ASTRAPlane Deployment
- Customer Control: Deployed via CloudFormation template in customer AWS account
- Managed Update: ASTRAflow provides updates; customer controls deployment timing
- Infrastructure as Code: All resources defined in Terraform/CloudFormation
- Monitoring: Automatic CloudWatch dashboards and alarms
- Scaling: Auto-scaling policies configured based on workload patterns

Zeebe Deployment
- Kubernetes: EKS cluster with multi-AZ redundancy
- HA Configuration: 3 brokers per partition, 3 partitions minimum
- Storage: EBS volumes with gp3 performance settings
- Backup: Automated daily snapshots retained for 30 days
- Update Strategy: Rolling updates with no downtime

9.2 Monitoring and Observability

Key Metrics
- System Health: CPU, memory, network utilization
- Application Performance: API latency, error rates, throughput
- Workflow Execution: Instance count, success rate, execution time
- Connector Performance: Connector-specific latency and error rates
- Data Pipeline: Audit trail ingestion rate, lag
- Cost: Resource utilization vs. billing projections

Alerting Strategy
- Operational Alerts: Sent to ASTRAflow operations team
- Customer Alerts: Customers alerted on their tenant-specific metrics
- Escalation: Critical incidents escalate to on-call engineers
- Runbooks: Automated playbooks for common issues

9.3 Disaster Recovery

Recovery Time Objective (RTO): 1 hour
Recovery Point Objective (RPO): 15 minutes
- Backup Strategy: Continuous replication with incremental backups every 15 minutes
- Multi-Region Failover: Automatic failover to standby region on primary region outage
- Data Retention: 30-day backup retention with archival to Glacier
- Recovery Testing: Quarterly disaster recovery drills
- Runbooks: Documented procedures for manual recovery scenarios

9.4 Performance Optimization
- Database Optimization: Query optimization, indexing, partitioning strategies
- Caching Strategy: Multi-layer caching (CDN, application, database)
- Connector Optimization: Connection pooling, batch operations
- Async Processing: Asynchronous workflows for long-running operations
- Resource Right-Sizing: Continuous monitoring to match resources to demand

---

10. Compliance and Data Residency

10.1 Data Residency

- Control Plane: Deployed in primary region (US or EU based on selection)
- Execution Data: Audit trails stored exclusively in customer-selected AWS region
- Cross-Border Data: Encrypted transit with customer-approved routing
- Data Localization: Customer data never leaves specified region without consent

10.2 Compliance Frameworks

- SOC 2 Type II: Annual attestation with 24-month observation period
- GDPR: Data processing agreements, rights management, breach notification
- HIPAA: Business associate agreements, encryption, audit controls
- PCI DSS: Secure handling of payment card data in workflows
- ISO 27001: Information security management system certification
- FedRAMP: Optional for government customers (when deployed in FedRAMP environments)

10.3 Data Subject Rights

- Right to Access: Customers can export all data within 30 days
- Right to Erasure: Data can be securely deleted with 90-day retention for compliance
- Right to Rectification: Data corrections logged with audit trail
- Data Portability: Export in standard formats (JSON, Parquet)

---

11. Technology Stack

11.1 Core Technologies

- Workflow Engine: Camunda 8 Zeebe 8.3+
- Language: Java 21 LTS
- Framework: Spring Boot 3.x
- API: REST (Spring MVC) + gRPC (Zeebe protocol)
- Frontend: React 18+ with TypeScript
- Database: PostgreSQL 15+ with read replicas
- Message Broker: Apache Kafka (for async events)
- Cache: Redis 7+ for sessions and rate limiting
- Container Runtime: Docker with multi-stage builds
- Orchestration: Kubernetes (EKS recommended)
- IaC: Terraform or AWS CloudFormation

11.2 AWS Services

- Compute: ECS Fargate, EKS, Lambda
- Storage: S3, EBS (with encryption)
- Database: RDS PostgreSQL, DynamoDB (optional)
- Networking: VPC, VPC Peering, AWS PrivateLink, PrivateDNS
- Security: IAM, Secrets Manager, KMS, ACM
- Monitoring: CloudWatch, X-Ray, CloudTrail
- Identity: Cognito (optional), or external OIDC provider
- CI/CD: CodePipeline, CodeBuild, CodeDeploy
- Configuration: Systems Manager Parameter Store, AppConfig

11.3 Connector Technologies

- HTTP Connector: REST API invocation with authentication

- AWS Connector: S3, Lambda, SNS, SQS, RDS integration
- Database Connectors: PostgreSQL, MySQL, Oracle, SQL Server
- Message Queues: Kafka, RabbitMQ, AWS SQS/SNS
- Git Integration: GitHub, GitLab, Bitbucket for CI/CD triggers
- Incident Management: PagerDuty, ServiceNow, Jira
- Monitoring: Datadog, New Relic, Prometheus exporters

---

12. API Design

12.1 Control Plane APIs

ASTRAStudio APIs

POST /api/v1/workflows
GET /api/v1/workflows/{workflowId}
PUT /api/v1/workflows/{workflowId}
DELETE /api/v1/workflows/{workflowId}
GET /api/v1/workflows/{workflowId}/executions
GET /api/v1/workflows/{workflowId}/executions/{executionId}
GET /api/v1/workflows/{workflowId}/audit

ASTRADeploy APIs

POST /api/v1/deployments
GET /api/v1/deployments/{deploymentId}
GET /api/v1/deployments/{deploymentId}/status
POST /api/v1/deployments/{deploymentId}/rollback

ASTRATenants APIs

POST /api/v1/tenants
GET /api/v1/tenants/{tenantId}
PUT /api/v1/tenants/{tenantId}/config
POST /api/v1/tenants/{tenantId}/users
GET /api/v1/tenants/{tenantId}/users
DELETE /api/v1/tenants/{tenantId}/users/{userId}

ASTRA Action Suite APIs

POST /api/v1/actions
GET /api/v1/actions/{actionId}
POST /api/v1/actions/{actionId}/build-image
GET /api/v1/actions/{actionId}/catalog

12.2 API Security
- Authentication: JWT bearer token in Authorization header
- Authorization: Tenant ID validated against user's assigned tenants
- Rate Limiting: 1000 requests/minute per API key
- Versioning: API version in URL path (/api/v1/, /api/v2/)
- Error Handling: Standardized error responses with correlation IDs

- **Webhook Support: Event-driven callbacks for async operations**

---

13. Integration Patterns

13.1 Workflow to External Systems

Synchronous Integration (Request-Reply)
- **HTTP service task calls REST API**
- **Wait for response (timeout configurable, default 30 seconds)**
- **Continue on success or error handler**

Asynchronous Integration (Fire-and-Forget)
- **Publish message to SNS/SQS**
- **Workflow continues immediately**
- **Optional: Callback connector waits for reply on dedicated queue**

Event-Driven Integration
- **Workflow publishes event to event bus (EventBridge)**
- **Other systems subscribe and react**
- **Optional: Workflow awaits event via catch event**

13.2 External Systems to ASTRAflow

Webhook Integration
- **External system calls ControlCenter webhook**
- **Webhook triggers workflow based on configuration**
- **Example: GitHub push event triggers deployment workflow**

Event Bridge Integration
- **Events published to AWS EventBridge**
- **EventBridge rules route events to ControlCenter**
- **ControlCenter evaluates alert-to-workflow mapping**

CloudWatch Integration
- **CloudWatch alarms trigger SNS topics**
- **SNS routes to ControlCenter webhook**
- **Workflow executes in response to alarm**

---

14. Future Enhancements

14.1 Roadmap
- **Multi-Region Zeebe: Active-active deployment across regions**
- **Advanced Analytics: Workflow optimization recommendations via ML**
- **Low-Code Connectors: Visual connector builder without coding**
- **Workflow Templates: Pre-built workflow templates for common use cases**
- **GitOps Integration: Version workflows in Git, deploy via CI/CD**
- **Simulator: Test workflows at scale before production deployment**
- **Conditional Deployment: Deploy workflows based on traffic patterns or time windows**
- **Hybrid Cloud: Support for on-premise Zeebe cluster deployments**

14.2 Performance Improvements
- **Zeebe 9.0+: Adoption of latest Zeebe performance optimizations**
- **gRPC Optimization: Connection pooling and HTTP/2 multiplexing**

- **Caching Strategy: Enhanced caching for workflow definitions and tenant configs**
- **Query Optimization: Database query optimization for execution history**

---

15. Glossary and Definitions

Tenant: A distinct business unit or application scope with isolated workflows, users, and AWS resources.
Workflow: A BPMN 2.0 process definition that automates a business process or alert response.
Execution: An instance of a workflow execution with unique execution ID and variables.
Connector: A reusable integration component that executes service tasks to call external systems.
Service Task: A workflow task that invokes a connector to perform work.
Audit Trail: Immutable log of all workflow executions and user actions.
STS Token: Temporary AWS credentials issued by Security Token Service with limited duration.
Split-Plane Architecture: Separation of control/design functions (SaaS) from execution functions (BYOC).
BYOC: Bring-Your-Own-Cloud model where execution runs in customer's AWS account.
Multi-Tenancy: Single platform instance serving multiple independent customers with isolated data.
Logical Isolation: Tenant separation at database level through row-level security.

Conclusion

ASTRAflow represents a modern approach to workflow automation by combining the ease of a managed SaaS platform with the security and control of customer-hosted execution environments. The Split-Plane Architecture ensures that organizations can leverage powerful automation capabilities while maintaining strict control over sensitive workflows and data residency.
By leveraging Camunda 8 Zeebe's proven multi-tenant architecture, AWS's comprehensive security and compliance capabilities, and thoughtful separation of concerns, ASTRAflow delivers enterprise-grade workflow orchestration suitable for mission-critical alert responses, business continuity planning, and complex automation scenarios.
The platform's commitment to security, compliance, and operational excellence makes it an ideal choice for organizations requiring both agility in workflow design and rigor in execution governance.

Document Version: 1.0
Last Updated: December 3, 2025
Status: Architecture Review Complete