Case Management Domain

Service Replica Configuration Report

Datadog Case Management Team

Generated: January 16, 2026

Executive Summary

This report provides a comprehensive overview of replica configurations for all services in the Case Management domain. The domain consists of 10 Kubernetes-deployed services with consistent replica configurations across all datacenters (us1, us3, us5, eu1, ap1, ap2) and environments (production and staging).

Key Findings:
• No datacenter-specific replica overrides detected
• All services use base replica counts from values.yaml
• Two services use horizontal pod autoscaling
• Multi-tenant services: case-api and case-event-handler

Service Replica Configuration Overview

Service Name	Base Replicas	Autoscaling	Tenants	Clusters/DCs
case-api	2	No	case-management on-call	us1, us3, us5, eu1, ap1, ap2, staging
case-intake	1	No	default	us1, us3, us5, eu1, ap1, ap2, staging
case-event-relay	1	No	case-management on-call	us1, us3, us5, eu1, ap1, ap2, staging
case-event-handler	2	Yes (min=2, max=10)	case-management on-call	us1, us3, us5, eu1, ap1, ap2, staging

Service Name	Base Replicas	Autoscaling	Tenants	Clusters/DCs
case-indexer	2	No	default	us1, us3, us5, eu1, ap1, ap2, staging
case-third-party-reader	1	No	default	us1, us3, us5, eu1, ap1, ap2, staging
case-event-feed-handler	2	Yes (min=2, max=10)	default (with flavors)	us1, us3, us5, eu1, ap1, ap2, staging
case-chat-interactions	1	No	default	us1, us3, us5, eu1, ap1, ap2, staging
case-synthetics	1	No	default	us1, us3, us5, eu1, ap1, ap2, staging
case-cli	1	No	default	us1, us3, us5, eu1, ap1, ap2, staging

Detailed Replica Configuration by Datacenter

The following table shows the replica count for each service across all production datacenters and staging environment. Note that **no datacenter-specific overrides** were found in the configuration files, meaning all datacenters use the base replica count defined in each service's config/k8s/values.yaml file.

Service	us1 prod	us3 prod	us5 prod	eu1 prod	ap1 prod	ap2 prod	us1 staging
case-api	2	2	2	2	2	2	2
case-intake	1	1	1	1	1	1	1
case-event-relay	1	1	1	1	1	1	1
case-event-handler	2*	2*	2*	2*	2*	2*	2*
case-indexer	2	2	2	2	2	2	2
case-third-party-reader	1	1	1	1	1	1	1
case-event-feed-handler	2*	2*	2*	2*	2*	2*	2*
case-chat-interactions	1	1	1	1	1	1	1
case-synthetics	1	1	1	1	1	1	1
case-cli	1	1	1	1	1	1	1

*** Indicates services with autoscaling enabled.** These services use Horizontal Pod Autoscaler (HPA) with minimum replica count shown (min=2, max=10).

Multi-Tenant Service Configuration

Two services support multi-tenancy with separate deployments for the **case-management** and **on-call** tenants:

1. case-api: Core gRPC server
• case-management tenant: 2 replicas per datacenter
• on-call tenant: 2 replicas per datacenter

2. case-event-handler: Kafka consumer with multiple flavors
• case-management tenant: 2-10 replicas (autoscaled) per datacenter
• on-call tenant: 2-10 replicas (autoscaled) per datacenter

Each tenant has separate Kafka topics and database schemas while sharing the same service code and deployment infrastructure.

Deployment Architecture Notes

Service Deployment Pattern:

All services follow Datadog's standard deployment patterns:
• **Runtime:** Go 1.21+
• **Build System:** Bazel
• **Orchestration:** Kubernetes with Helm charts
• **Health Checks:** Liveness/readiness probes on port 8080
• **Observability:** dd-trace (APM), StatsD (metrics), structured logging

Autoscaling Configuration:

Two services use Horizontal Pod Autoscaler (HPA):
• **case-event-handler:** Scales based on Kafka consumer lag
  - Minimum replicas: 2
  - Maximum replicas: 10
  - Supports 12 different event handler flavors

• **case-event-feed-handler:** Scales based on event processing load
  - Minimum replicas: 2
  - Maximum replicas: 10
  - Processes domain events for the event feed

Active-Passive Services:

case-event-relay uses a distributed locking pattern for active-passive deployment:
• Only 1 replica configured per datacenter
• Uses PostgreSQL-based distributed lock
• Automatic failover within 20-30 seconds
• Implements transactional outbox pattern for reliable event delivery

Configuration Management:

Service configurations are organized hierarchically:
• **Base configuration:** config/k8s/values.yaml
• **Tenant configuration:** config/k8s/values/tenants/{tenant}/
• **Datacenter configuration:** config/k8s/values/tenants/{tenant}/{dc}.yaml

Replica counts are defined at the base level and can be overridden at tenant or datacenter levels, though **no overrides exist in the current configuration**.

Configuration Consistency

The Case Management domain demonstrates excellent configuration consistency:

✓ **Uniform Replica Counts:** All datacenters use the same replica configuration
✓ **No Special Cases:** No datacenter-specific overrides detected
✓ **Clear Scaling Strategy:** Fixed replicas for stateful services, autoscaling for event processors
✓ **High Availability:** Multi-replica deployments for critical services
✓ **Resource Efficiency:** Single replicas for non-critical services

This uniform configuration simplifies operations, reduces the risk of configuration drift, and ensures consistent behavior across all production environments.

This report was generated from the Kubernetes configuration files in domains/case_management/apps//config/k8s/. For the latest configuration details, refer to the source code in the dd-source repository.*