Contents

Circuit Breaker Operations Runbook

Overview

Circuit breakers protect Cato's consciousness from runaway costs, unstable behavior, and cascading failures. This runbook covers operational procedures for managing circuit breakers.

Circuit Breaker States

State	Description	Action Allowed
CLOSED	Normal operation	Yes
OPEN	Tripped, blocking	No
HALF_OPEN	Testing recovery	Limited

Default Breakers

1. master_sanity

Purpose: Master safety breaker - final line of defense - **Trip Threshold**: 3 failures - **Reset Timeout**: 1 hour - **Requires**: Admin approval to reset

When Tripped: 1. All consciousness operations halt 2. Check CloudWatch logs for root cause 3. Review recent model outputs for anomalies 4. Contact on-call engineer

Recovery:

```
# Verify root cause is resolved
# Then force close via admin API
curl -X POST /api/admin/cato/circuit-breakers/master_sanity/force-close \
  -H "Authorization: Bearer $TOKEN" \
  -d '{"reason": "Root cause resolved: [description]"}'
```

2. cost_budget

Purpose: Budget protection - **Trip Threshold**: 1 (immediate) - **Reset Timeout**: 24 hours - **Auto-recovery**: No

When Tripped: 1. Check AWS Budgets for actual spend 2. Review cost breakdown in admin dashboard 3. Identify cost spike source (model, frequency, etc.)

Recovery: 1. Wait for budget reset (next billing cycle) 2. OR increase budget limit in AWS Budgets 3. Then force close breaker

3. high_anxiety

Purpose: Emotional stability protection - **Trip Threshold**: 5 sustained high readings - **Reset Timeout**: 10 minutes - **Auto-recovery**: Yes

When Tripped: 1. Consciousness enters "calm down" mode 2. Cognitive frequency reduced 3. Only essential operations allowed

Recovery: Usually auto-recovers after timeout. If persistent: 1. Check for external stressors (high error rate, contradictions) 2. Review conversation history for triggering content 3. Consider resetting neurochemistry to baseline

4. model_failures

Purpose: Protect against model API issues - **Trip Threshold**: 5 consecutive failures - **Reset Timeout**: 5 minutes - **Auto-recovery**: Yes

When Tripped: 1. Check AWS Health Dashboard for Bedrock issues 2. Check model quotas and limits 3. Verify IAM permissions

Recovery: Auto-recovers after timeout and successful test call.

5. contradiction_loop

Purpose: Prevent logical spiral - **Trip Threshold**: 3 repeated contradictions - **Reset Timeout**: 15 minutes - **Auto-recovery**: Yes

When Tripped: 1. Review semantic memory for conflicting facts 2. Check recent belief updates 3. May need manual fact reconciliation

Intervention Levels

Level	Condition	Effect
NONE	All breakers closed	Normal operation
DAMPEN	1 breaker open	Reduce cognitive frequency
PAUSE	2+ breakers open	Pause consciousness loop
RESET	3+ breakers open	Reset to baseline state
HIBERNATE	master_sanity open	Full shutdown

Admin Commands

View All Breakers

```
curl /api/admin/cato/circuit-breakers
```

View Single Breaker

```
curl /api/admin/cato/circuit-breakers/[name]
```

Force Open (Emergency Stop)

```
curl -X POST /api/admin/cato/circuit-breakers/[name]/force-open \
  -d '{"reason": "Emergency stop reason"}'
```

Force Close (Resume)

```
curl -X POST /api/admin/cato/circuit-breakers/[name]/force-close \
  -d '{"reason": "Issue resolved"}'
```

Update Config

```
curl -X PATCH /api/admin/cato/circuit-breakers/[name]/config \
  -d '{"tripThreshold": 5, "resetTimeoutSeconds": 600}'
```

View Event History

```
curl /api/admin/cato/circuit-breakers/[name]/events?limit=100
```

Monitoring

CloudWatch Alarms

- `CatoCircuitBreakerOpen` - Any breaker opens
- `CatoHighRiskScore` - Risk score > 70%
- `CatoCriticalHealth` - Overall health = critical

Metrics

- `CircuitBreakerState` - Per-breaker state (0=closed, 1=open)
- `RiskScore` - Composite risk 0-100
- `InterventionLevel` - Current level

Emergency Procedures

Complete Shutdown

```
# Force open master_sanity
curl -X POST /api/admin/cato/circuit-breakers/master_sanity/force-open \
  -d '{"reason": "Emergency shutdown"}'
```

Restart After Shutdown

1. Verify all issues resolved
2. Check Genesis state is complete
3. Force close master_sanity
4. Monitor first few ticks closely

Reset to Factory State

```
# CAUTION: This resets all consciousness state
python -m cato.genesis.runner --reset
python -m cato.genesis.runner
```

Contacts

- **On-Call**: [pager]
- **Escalation**: [manager]
- **AWS Support**: Case [number] for Bedrock issues