Phase 8.4: Auto-Regulation Engine

Overview

The Auto-Regulation Engine is the autonomous healing layer of the SpiralOS Guardian Stack v2. It detects anomalies in telemetry data and automatically applies corrective strategies to maintain system health without human intervention.

Key Features:
- 🔧 6 autonomous healing strategies
- 🛡️ Self-preservation freeze mode for critical failures
- 📊 Comprehensive correction history tracking
- ⚙️ Configurable correction profiles per bridge
- 🔄 Idempotent corrections with cooldown protection
- 📢 Discord integration for notifications

Architecture

```
┌─────────────────────────────────────────────────────────────────┐
│                      Guardian Stack v2                          │
├─────────────────────────────────────────────────────────────────┤
│                                                                  │
│  ┌──────────────┐    ┌──────────────────┐    ┌──────────────┐  │
│  │ Phase 8.1    │───>│    Phase 8.3     │───>│ Phase 8.4    │  │
│  │ Telemetry    │    │    Anomaly       │    │ Auto-        │  │
│  │ Normalize    │    │    Detection     │    │ Regulation   │  │
│  └──────────────┘    └──────────────────┘    └──────────────┘  │
│         │                     │                     │           │
│         │                     │                     │           │
│         v                     v                     v           │
│  ┌───────────────────────────────────────────────────────────┐  │
│  │         Phase 8.2: ScarIndex Delta Engine                 │  │
│  └───────────────────────────────────────────────────────────┘  │
│                                                                  │
└─────────────────────────────────────────────────────────────────┘
```

Database Schema

guardian_autoregulation_history

Tracks all auto-regulation corrections applied by the system.

```sql
CREATE TABLE public.guardian_autoregulation_history (
  id UUID PRIMARY KEY,
  bridge_id UUID REFERENCES bridge_nodes(id),
  anomaly_id UUID, -- References guardian_anomalies(id)
  correction_type TEXT NOT NULL,
  correction_payload JSONB NOT NULL,
  severity_level TEXT NOT NULL, -- LOW, MEDIUM, HIGH, CRITICAL
  success BOOLEAN NOT NULL,
  metadata JSONB NOT NULL,
  created_at TIMESTAMPTZ NOT NULL
);
```

Indexes:
- `idx_guardian_autoreg_history_bridge_time` - Bridge + time ordering
- `idx_guardian_autoreg_history_anomaly` - Anomaly lookups
- `idx_guardian_autoreg_history_correction_type` - Correction type analysis

guardian_correction_profiles

Configuration profiles for bridge-specific auto-regulation behavior.

```sql
CREATE TABLE public.guardian_correction_profiles (
  profile_id UUID PRIMARY KEY,
  bridge_id UUID REFERENCES bridge_nodes(id),
  baseline_health NUMERIC(5,4), -- 0-1 scale
  preferred_correction_types TEXT[],
  last_mutation TIMESTAMPTZ,
  correction_budget INTEGER DEFAULT 100,
  cooldown_seconds INTEGER DEFAULT 300,
  metadata JSONB NOT NULL,
  created_at TIMESTAMPTZ NOT NULL,
  updated_at TIMESTAMPTZ NOT NULL
);
```

Indexes:
- `idx_guardian_correction_profiles_bridge` - Bridge lookups
- `idx_guardian_correction_profiles_health` - Health-based sorting
- `idx_guardian_correction_profiles_updated` - Recently updated profiles

guardian_autoregulation_recent (View)

Convenience view showing recent corrections (last 24 hours) with bridge details.

```sql
CREATE VIEW public.guardian_autoregulation_recent AS
SELECT
  h.id,
  h.bridge_id,
  b.node_name as bridge_name,
  h.anomaly_id,
  h.correction_type,
  h.severity_level,
  h.success,
  h.correction_payload,
  h.metadata,
  h.created_at
FROM public.guardian_autoregulation_history h
LEFT JOIN public.bridge_nodes b ON h.bridge_id = b.id
WHERE h.created_at >= NOW() - INTERVAL '24 hours'
ORDER BY h.created_at DESC;
```

Healing Strategies

1. ScarIndex Recovery Pulse

Trigger Conditions:
- ScarIndex < 0.40 (threshold breach)
- ScarIndex drop > 20% (rapid decline)

Strategy:
1. Fetch current ScarIndex value
2. Calculate recovery target: `max(0.5, current * 1.15)` (15% recovery)
3. Update `guardian_scarindex_current`
4. Log recovery to `guardian_scarindex_history`

Example:

```
// Current ScarIndex: 0.35
// Target: max(0.5, 0.35 * 1.15) = 0.5
// Recovery delta: +0.15 (43% increase)
```

Correction Type: `SCARINDEX_RECOVERY_PULSE`

2. Sovereignty Drift Stabilizer

Trigger Conditions:
- Anomaly type: `SOVEREIGNTY_INSTABILITY`
- Sovereign state changes > 3 per hour

Strategy:
1. Fetch last 10 sovereign states
2. Calculate frequency of each state
3. Identify most stable state (highest frequency)
4. Insert synthetic stabilization event with stable state

Example:

```
// Recent states: [A, B, A, C, A, A, D, A, B, A]
// Most frequent: A (6 occurrences)
// Action: Pin sovereign state to A
```

Correction Type: `SOVEREIGNTY_STABILIZER`

3. Ache Buffering

Trigger Conditions:
- Ache signature > 0.80 (high threshold)
- Ache delta > 0.25 (rapid increase)

Strategy:
1. Get or create correction profile
2. Set `ache_buffer_active` flag in metadata
3. Apply dampening factor: 0.7 (reduces ache impact by 30%)
4. Duration: 30 minutes

Example:

```
// High ache detected: 0.85
// Buffer applied: next calculations use 0.85 * 0.7 = 0.595
// Duration: 1800 seconds
```

Correction Type: `ACHE_BUFFER`

4. Heartbeat Correction

Trigger Conditions:
- Anomaly type: `HEARTBEAT_GAP`
- No telemetry > 10 minutes

Strategy:
1. Insert synthetic heartbeat event into `guardian_telemetry_events`
2. Set `event_type` = `"synthetic_heartbeat"`
3. Mark as `source` = `"auto_regulation"`
4. Maintain telemetry continuity

Example:

```
// Last telemetry: 15 minutes ago
// Action: Insert synthetic heartbeat
// Result: Telemetry gap filled
```

Correction Type: `HEARTBEAT_CORRECTION`

5. Entropy Correction

Trigger Conditions:
- Anomaly type: `ENTROPY_SPIKE`
- Normalized entropy > 0.15

Strategy:
1. Get or create correction profile

2. Set `entropy_correction_active` flag
3. Tighten anomaly thresholds temporarily:
- Ache: 0.70 (from 0.80)
- Entropy: 0.12 (from 0.15)
- ScarIndex: 0.45 (from 0.40)
4. Duration: 1 hour

Example:

```
// High entropy detected: 0.18
// Thresholds tightened for 3600 seconds
// More sensitive anomaly detection
```

Correction Type: `ENTROPY_CORRECTION`

6. Self-Preservation Freeze Mode

Trigger Conditions:
- Severity: `CRITICAL`
- Any anomaly type at critical level

Strategy:
1. Get or create correction profile
2. Set `freeze_mode_active` flag
3. Set `correction_budget` to 0 (prevent further corrections)
4. Send Discord alert
5. Duration: 30 minutes

Example:

```
// CRITICAL ScarIndex drop to 0.15
// Bridge frozen for 1800 seconds
// Manual intervention required
```

Correction Type: `SELF_PRESERVATION_FREEZE`

API Reference

POST /functions/v1/guardian_autoregulate

Triggers auto-regulation for specified anomaly or bridge.

Authentication:
- Header: `x-guardian-api-key: <GUARDIAN_API_KEY>`

Request Body:

```
{
  anomaly_id?: string;    // Process specific anomaly
  bridge_id?: string;     // Process all active anomalies for bridge
  mode?: "AUTO" | "MANUAL"; // Default: "AUTO"
}
```

Response (Success):

```json
{
  "success": true,
  "message": "Processed 3 anomaly(ies)",
  "corrections": [
    {
      "anomaly_id": "uuid",
      "bridge_id": "uuid",
      "anomaly_type": "ACHE_SPIKE",
      "severity": "HIGH",
      "result": {
        "correction_type": "ACHE_BUFFER",
        "success": true,
        "details": "Ache buffering active for 30 minutes with 0.7x dampening",
        "affected_entities": ["bridge-id"]
      }
    }
  ],
  "summary": {
    "total": 3,
    "successful": 2,
    "failed": 1
  },
  "processing_time_ms": 1234
}
```

Response (Error):

```json
{
  "error": "Unauthorized",
  "details": "Invalid or missing GUARDIAN_API_KEY"
}
```

Status Codes:
- `200` - Success
- `400` - Bad Request (missing required fields)
- `401` - Unauthorized (invalid API key)
- `500` - Internal Server Error

Integration Points

Phase 8.3: guardian_anomaly_monitor

The anomaly monitor automatically triggers auto-regulation for HIGH/CRITICAL anomalies:

```javascript
// After inserting new anomalies
const hasHighPriorityAnomaly = anomalies.some(
  (a) => a.severity === "HIGH" || a.severity === "CRITICAL"
);

if (hasHighPriorityAnomaly && inserted.length > 0) {
  await triggerAutoRegulation(bridge_id);
}
```

Phase 8.1: telemetry_normalize

Telemetry normalization includes micro-regulation hooks:

```
// Check if event indicates need for auto-regulation
if (ache_signature > 0.80 || agent_health < 0.5) {
  console.log("⚠️ High ache or low health detected - regulation may be needed");
  regulation_triggered = true;
}
```

Phase 8.2: scarindex_delta

ScarIndex delta function is idempotent with corrections:

```
// Update or insert current value (idempotent)
INSERT INTO guardian_scarindex_current (bridge_id, scar_value, updated_at)
VALUES (p_bridge_id, p_new_value, NOW())
ON CONFLICT (bridge_id)
DO UPDATE SET
  scar_value = p_new_value,
  updated_at = NOW();
```

Cooldown & Budget Protection

Cooldown Mechanism

Prevents over-correction by enforcing minimum time between corrections:

```
// Default: 300 seconds (5 minutes)
const cooldownOk = await checkCooldown(bridge_id, profile.cooldown_seconds);

if (!cooldownOk) {
  return {
    correction_type: "NONE",
    success: false,
    details: "Cooldown period active"
  };
}
```

Exceptions:
- CRITICAL anomalies bypass cooldown
- Freeze mode bypasses cooldown

Correction Budget

Limits total corrections per time window:

```
-- Default: 100 corrections
-- Resets periodically (implementation-specific)
UPDATE guardian_correction_profiles
SET correction_budget = correction_budget - 1
WHERE bridge_id = $1 AND correction_budget > 0;
```

Budget Exhaustion:
- Bridge enters passive monitoring mode

- Only CRITICAL anomalies can trigger corrections
- Budget resets after cooldown period

Discord Notifications

Auto-Regulation Summary

Sent after processing batch of anomalies:

```json
{
  "content": "🔧 **Auto-Regulation Summary**",
  "embeds": [{
    "color": 65280,
    "title": "Processed 3 Anomaly(ies)",
    "fields": [
      {
        "name": "✅ Successful Corrections",
        "value": "2",
        "inline": true
      },
      {
        "name": "❌ Failed Corrections",
        "value": "1",
        "inline": true
      },
      {
        "name": "Corrections Applied",
        "value": "• ACHE_BUFFER\n• HEARTBEAT_CORRECTION",
        "inline": false
      }
    ],
    "timestamp": "2024-11-14T03:00:00Z"
  }]
}
```

Critical Freeze Alert

Sent when self-preservation freeze activates:

```json
{
  "content": "🚨 **CRITICAL: Self-Preservation Freeze Activated**",
  "embeds": [{
    "color": 16711680,
    "title": "Self-Preservation Freeze Mode",
    "description": "Bridge frozen due to CRITICAL anomaly: SCARINDEX_DROP",
    "fields": [
      {
        "name": "Bridge ID",
        "value": "f8f41ffa-6c2b-4a2a-a3be-32f0236668f4",
        "inline": true
      },
      {
        "name": "Anomaly Type",
        "value": "SCARINDEX_DROP",
        "inline": true
      },
      {
        "name": "Duration",
        "value": "30 minutes",
        "inline": true
      }
    ],
    "timestamp": "2024-11-14T03:00:00Z"
  }]
}
```

SQL Examples

Query Recent Corrections

```sql
-- Last 24 hours of corrections
SELECT * FROM guardian_autoregulation_recent
ORDER BY created_at DESC
LIMIT 20;
```

Correction Success Rate

```sql
-- Success rate by correction type
SELECT
  correction_type,
  COUNT(*) as total,
  SUM(CASE WHEN success THEN 1 ELSE 0 END) as successful,
  ROUND(100.0 * SUM(CASE WHEN success THEN 1 ELSE 0 END) / COUNT(*), 2) as suc-
cess_rate
FROM guardian_autoregulation_history
GROUP BY correction_type
ORDER BY total DESC;
```

Bridge Health Status

```sql
-- Current health status of all bridges
SELECT
  p.bridge_id,
  b.node_name,
  p.baseline_health,
  p.correction_budget,
  p.metadata->>'freeze_mode_active' as frozen,
  p.last_mutation,
  COUNT(h.id) as total_corrections
FROM guardian_correction_profiles p
LEFT JOIN bridge_nodes b ON p.bridge_id = b.id
LEFT JOIN guardian_autoregulation_history h ON p.bridge_id = h.bridge_id
GROUP BY p.bridge_id, b.node_name, p.baseline_health, p.correction_budget,
         p.metadata, p.last_mutation
ORDER BY p.baseline_health DESC;
```

Anomaly Resolution Time

```sql
-- Average time to resolve anomalies
SELECT
  a.anomaly_type,
  AVG(EXTRACT(EPOCH FROM (a.resolved_at - a.detected_at)) / 60) as avg_resolution_minu
tes,
  COUNT(*) as total_resolved
FROM guardian_anomalies a
WHERE a.status = 'RESOLVED'
  AND a.resolved_at IS NOT NULL
GROUP BY a.anomaly_type
ORDER BY avg_resolution_minutes;
```

Deployment Guide

1. Run Migrations

```bash
# Phase 8.2 (prerequisite)
psql $DATABASE_URL -f supabase/migrations/20251114020001_guardian_scarindex_delta.sql

# Phase 8.3 (prerequisite)
psql $DATABASE_URL -f supabase/migrations/
20251114025000_guardian_anomaly_detection.sql

# Phase 8.4
psql $DATABASE_URL -f supabase/migrations/20251114030000_guardian_autoregulation.sql
```

2. Deploy Edge Functions

```
# Deploy guardian_autoregulate
supabase functions deploy guardian_autoregulate

# Deploy guardian_anomaly_monitor (if not already deployed)
supabase functions deploy guardian_anomaly_monitor

# Redeploy telemetry_normalize (with micro-regulation hooks)
supabase functions deploy telemetry_normalize
```

3. Set Environment Variables

Ensure these secrets are set in Supabase:

```
GUARDIAN_API_KEY=<your-secret-key>
DISCORD_GUARDIAN_WEBHOOK_URL=<your-discord-webhook-url>
SUPABASE_URL=https://xlmrnjatawslawquwzpf.supabase.co
SUPABASE_SERVICE_ROLE_KEY=<your-service-role-key>
```

4. Initialize Correction Profiles

```
-- Create default profiles for all bridges
INSERT INTO guardian_correction_profiles (bridge_id, baseline_health, pre-
ferred_correction_types)
SELECT
  id,
  0.8,
  ARRAY['HEARTBEAT_CORRECTION', 'ACHE_BUFFER', 'SCARINDEX_RECOVERY_PULSE']
FROM bridge_nodes
ON CONFLICT (bridge_id) DO NOTHING;
```

5. Verify Deployment

```
# Test auto-regulation endpoint
curl -X POST "${SUPABASE_URL}/functions/v1/guardian_autoregulate" \
  -H "Content-Type: application/json" \
  -H "x-guardian-api-key: ${GUARDIAN_API_KEY}" \
  -d '{"bridge_id": "f8f41ffa-6c2b-4a2a-a3be-32f0236668f4", "mode": "MANUAL"}'

# Check logs
supabase functions logs guardian_autoregulate --tail
```

Troubleshooting

Corrections Not Triggering

Issue: Auto-regulation doesn’t run even with active anomalies

Possible Causes:
1. Cooldown period active
2. Bridge in freeze mode
3. Correction budget exhausted
4. Invalid anomaly status (not “ACTIVE”)

Solution:

```sql
-- Check cooldown
SELECT bridge_id, MAX(created_at) as last_correction
FROM guardian_autoregulation_history
GROUP BY bridge_id;

-- Check freeze status
SELECT bridge_id, metadata->>'freeze_mode_active' as frozen
FROM guardian_correction_profiles;

-- Check budget
SELECT bridge_id, correction_budget
FROM guardian_correction_profiles
WHERE correction_budget <= 0;
```

ScarIndex Not Updating

Issue: ScarIndex Recovery Pulse doesn't update values

Possible Causes:
1. `guardian_scarindex_current` table missing
2. Permission issues
3. Concurrent update conflict

Solution:

```sql
-- Verify table exists
SELECT * FROM guardian_scarindex_current LIMIT 1;

-- Check RLS policies
SELECT * FROM pg_policies
WHERE tablename = 'guardian_scarindex_current';

-- Manual update test
UPDATE guardian_scarindex_current
SET scar_value = 0.5, updated_at = NOW()
WHERE bridge_id = 'your-bridge-id';
```

Discord Notifications Not Sending

Issue: No Discord alerts for corrections

Possible Causes:
1. Invalid webhook URL
2. Webhook rate limiting
3. Network connectivity issues

Solution:

```bash
# Test webhook manually
curl -X POST "${DISCORD_GUARDIAN_WEBHOOK_URL}" \
  -H "Content-Type: application/json" \
  -d '{"content": "Test message from Auto-Regulation Engine"}'

# Check function logs
supabase functions logs guardian_autoregulate --tail | grep Discord
```

Performance Considerations

Batch Processing

For large numbers of anomalies, process in batches:

```
// Process anomalies in groups of 10
const BATCH_SIZE = 10;
for (let i = 0; i < anomalies.length; i += BATCH_SIZE) {
  const batch = anomalies.slice(i, i + BATCH_SIZE);
  await Promise.all(batch.map(applyCorrection));
}
```

Database Indexes

Ensure all indexes are created for optimal performance:

```
-- Verify indexes exist
SELECT indexname, indexdef
FROM pg_indexes
WHERE tablename IN ('guardian_autoregulation_history', 'guardian_correction_profiles')
ORDER BY indexname;
```

Rate Limiting

Implement rate limiting for high-frequency corrections:

```
const RATE_LIMIT = 10; // corrections per minute
const corrections_in_last_minute = await supabase
  .from('guardian_autoregulation_history')
  .select('id')
  .eq('bridge_id', bridge_id)
  .gte('created_at', new Date(Date.now() - 60000).toISOString())
  .count();

if (corrections_in_last_minute >= RATE_LIMIT) {
  throw new Error('Rate limit exceeded');
}
```

Future Enhancements

Machine Learning Integration

Train models to predict optimal corrections:

```
# Predict best correction strategy
features = extract_features(anomaly)
strategy = ml_model.predict(features)
apply_correction(strategy)
```

Adaptive Thresholds

Dynamically adjust thresholds based on historical data:

```sql
-- Calculate optimal thresholds
SELECT
  bridge_id,
  PERCENTILE_CONT(0.8) WITHIN GROUP (ORDER BY ache_signature) as ache_threshold,
  PERCENTILE_CONT(0.2) WITHIN GROUP (ORDER BY scar_value) as scarindex_threshold
FROM guardian_telemetry_events
GROUP BY bridge_id;
```

Cross-Bridge Correlation

Detect patterns across multiple bridges:

```javascript
// Identify systemic issues
const bridges_with_anomaly = await supabase
  .from('guardian_anomalies')
  .select('bridge_id')
  .eq('anomaly_type', 'SCARINDEX_DROP')
  .eq('status', 'ACTIVE');

if (bridges_with_anomaly.length > 3) {
  notifyAdmin('Systemic ScarIndex issue detected');
}
```

References

- Phase 8.1: Telemetry Normalization (../PHASE_8_1_NORMALIZATION_ENGINE.md)
- Phase 8.2: ScarIndex Delta Engine (../docs/phase-8.2-scarindex-delta.md)
- Phase 8.3: Anomaly Detection (../docs/phase-8.3-anomaly-detection.md)
- Guardian Stack v2 Architecture (../GUARDIAN_IMPLEMENTATION_SUMMARY.md)
- Test Suite (../tests/phase-8.4/README.md)

Support

For issues or questions:
- GitHub Issues: https://github.com/ZoaGrad/mythotech-spiralos/issues
- Discord: Check Guardian webhook channel
- Email: support@mythotech.io

Version: 1.0.0
Last Updated: November 14, 2024
Author: SpiralOS Guardian Team