34 Technical Support Documentation

MWRASP Quantum Defense System

Generated: 2025-08-24 18:15:03

TOP SECRET//SCI - HANDLE VIA SPECIAL ACCESS CHANNELS

MWRASP Quantum Defense System - Technical Support Documentation

Enterprise Support Operations Manual

Version 4.0 | August 2025

EXECUTIVE SUMMARY

Support Framework Overview

- **Support Tiers**: 4-level escalation structure
- **Response SLA**: 15-minute P1, 1-hour P2, 4-hour P3
- **Resolution Rate**: 94% first-contact resolution
- **Customer Satisfaction**: 98.7% CSAT score

- **Coverage**: 24/7/365 global support with follow-the-sun model

Key Capabilities

- AI-powered issue prediction and prevention
- Quantum-secure remote diagnostics
- Automated remediation for 67% of issues
- Real-time system health monitoring
- Proactive incident prevention

---

1. SUPPORT TIER STRUCTURE

1.1 Tier 1 - Frontline Support

```
class Tier1Support:
   def responsibilities(self):
       return {
           'initial_contact': {
               'channels': ['Phone', 'Email', 'Chat', 'Portal'],
               'response_time': {
                   'phone': 'Immediate',
                   'chat': '<30 seconds',
                   'email': '<1 hour',
                   'portal': '<2 hours'
               },
               'coverage': '24/7/365 global'
           },

           'issue_categories': {
               'authentication': [
                   'Password resets',
                   'MFA troubleshooting',
                   'Account lockouts',
                   'Token refresh issues'
               ],

               'basic_configuration': [
                   'Agent deployment',
                   'Initial setup',
                   'Network connectivity',
                   'Certificate installation'
               ],
```

```
                'monitoring': [
                    'Dashboard access',
                    'Alert configuration',
                    'Report generation',
                    'Metric interpretation'
                ]
            },

            'resolution_tools': {
                'knowledge_base': '2,847 articles',
                'automated_scripts': 347,
                'diagnostic_tools': 89,
                'escalation_criteria': 'Defined matrix'
            },

            'metrics': {
                'first_contact_resolution': '67%',
                'average_handle_time': '8.3 minutes',
                'customer_satisfaction': '94%',
                'escalation_rate': '33%'
            }
        }

    def troubleshooting_scripts(self):
        return {
            'connectivity_issues': '''
                1. Verify network connectivity
                   - ping quantum-gateway.mwrasp.ai
                   - traceroute to endpoints
                   - Check firewall rules (ports 8443, 9443)

                2. Validate certificates
                   - openssl s_client -connect api.mwrasp.ai:443
                   - Check certificate expiration
                   - Verify CA trust chain

                3. Test API connectivity
                   - curl https://api.mwrasp.ai/v1/health
                   - Verify response code 200
                   - Check response time <100ms
            ''',

            'authentication_failures': '''
                1. Verify credentials
                   - Check username format
                   - Validate password complexity
                   - Confirm account status

                2. MFA troubleshooting
                   - Resync time-based tokens
                   - Verify authenticator app
                   - Check backup codes
```

```
                3. Session management
                   - Clear browser cache
                   - Reset quantum tokens
                   - Verify SSO configuration
            '''
        }
```

1.2 Tier 2 - Advanced Technical Support

```
tier2_support:
 expertise_areas:
   quantum_cryptography:
     - Key rotation procedures
     - Algorithm migration
     - Performance optimization
     - Entropy pool management

   ai_agent_management:
     - Behavioral calibration
     - Model updates
     - Performance tuning
     - Anomaly investigation

   system_integration:
     - API troubleshooting
     - SIEM integration
     - Identity provider configuration
     - Custom workflow development

 diagnostic_capabilities:
   remote_access:
     protocol: "Quantum-secure SSH"
     authentication: "Certificate + MFA"
     audit: "Full session recording"
     approval: "Customer consent required"

   log_analysis:
     tools:
       - Splunk Enterprise
       - ELK Stack
       - Custom ML analyzers
     retention: "90 days online"
     correlation: "AI-powered"

   performance_analysis:
     metrics:
       - Latency profiling
       - Throughput analysis
```

```
        - Resource utilization
        - Bottleneck identification

  escalation_criteria:
    to_tier3:
      - Code-level issues
      - Quantum algorithm problems
      - Critical security incidents
      - Performance degradation >20%

    response_sla:
      priority_1: "30 minutes"
      priority_2: "2 hours"
      priority_3: "8 hours"
      priority_4: "24 hours"
```

1.3 Tier 3 - Engineering Support

```
class Tier3Engineering:
    def specialized_support(self):
        return {
            'code_level_debugging': {
                'languages': ['Python', 'Go', 'Rust', 'C++'],
                'tools': [
                    'GDB with quantum extensions',
                    'Memory profilers',
                    'Race condition detectors',
                    'Quantum state analyzers'
                ],
                'access_level': 'Source code repository'
            },

            'patch_development': {
                'hotfix_process': {
                    'identification': 'Root cause analysis',
                    'development': '4-hour SLA for P1',
                    'testing': 'Automated regression suite',
                    'deployment': 'Blue-green with rollback'
                },

                'custom_patches': {
                    'approval': 'Engineering manager',
                    'testing': 'Customer UAT environment',
                    'support': '90-day warranty',
                    'merge': 'Next release candidate'
                }
            },

            'performance_optimization': {
```

```
                'profiling': {
                    'cpu': 'Intel VTune, AMD uProf',
                    'memory': 'Valgrind, AddressSanitizer',
                    'network': 'Wireshark with quantum decrypt',
                    'quantum': 'Custom qubit analyzers'
                },

                'optimization_techniques': [
                    'Algorithm refinement',
                    'Caching strategies',
                    'Parallel processing',
                    'Quantum circuit optimization'
                ]
            },

            'incident_command': {
                'p1 incidents': 'Engineering lead required',
                'war_room': 'Virtual command center',
                'communication': '15-minute updates',
                'post_mortem': 'Within 48 hours'
            }
        }
```

1.4 Tier 4 - Vendor/Development Team

```
tier4_development:
 engagement model:
   triggers:
     - Core platform bugs
     - Quantum algorithm issues
     - Security vulnerabilities
     - Feature requests

   communication:
     channel: "Dedicated Slack workspace"
     response time: "4 hours for P1"
     escalation: "VP Engineering"

   deliverables:
     bug fixes:
       timeline: "Sprint inclusion"
       testing: "Full regression"
       deployment: "Coordinated release"

     feature development:
       process: "Quarterly roadmap"
       prioritization: "Customer advisory board"
       delivery: "Agile sprints"
```

```
  knowledge_transfer:
    documentation:
      - Architecture deep dives
      - Algorithm specifications
      - Integration patterns
      - Best practices

    training:
      frequency: "Monthly tech talks"
      format: "Video + hands-on labs"
      certification: "Annual recertification"
```

---

2. INCIDENT MANAGEMENT PROCEDURES

2.1 Incident Classification

```
class IncidentClassification:
    def priority_matrix(self):
        return {
            'P1_Critical': {
                'definition': 'Complete system outage or security
breach',
                'impact': 'All agents affected',
                'response_sla': '15 minutes',
                'resolution_sla': '4 hours',
                'escalation': 'Immediate to Tier 3',
                'communication': 'Executive notification',
                'examples': [
                    'Quantum gateway failure',
                    'Mass agent disconnection',
                    'Data breach detected',
                    'Ransomware attack'
                ]
            },

            'P2_High': {
                'definition': 'Partial outage or degraded
performance',
                'impact': '>30% agents affected',
                'response_sla': '1 hour',
                'resolution_sla': '8 hours',
                'escalation': 'Tier 2 within 2 hours',
                'communication': 'Customer notification',
                'examples': [
                    'Regional gateway issues',
                    'API performance degradation',
                    'Authentication delays',
```

```
                    'Alerting system failure'
                ]
            },

            'P3_Medium': {
                'definition': 'Limited impact or workaround
available',
                'impact': '<30% agents affected',
                'response_sla': '4 hours',
                'resolution_sla': '24 hours',
                'escalation': 'As needed',
                'communication': 'Status page update',
                'examples': [
                    'Dashboard loading issues',
                    'Report generation delays',
                    'Non-critical feature failure',
                    'Documentation errors'
                ]
            },

            'P4_Low': {
                'definition': 'Minimal impact or cosmetic issues',
                'impact': 'Individual users',
                'response_sla': '24 hours',
                'resolution_sla': '5 days',
                'escalation': 'Standard process',
                'communication': 'Ticket update',
                'examples': [
                    'UI formatting issues',
                    'Feature requests',
                    'Documentation updates',
                    'Training questions'
                ]
            }
        }
```

2.2 Incident Response Workflow

```
def incident_response_workflow(incident_priority: str) -> dict:
    """
    Automated incident response orchestration
    """
    workflow = {
        'detection': {
            'sources': [
                'Monitoring alerts',
                'Customer reports',
                'AI anomaly detection',
                'Quantum canary triggers'
```

```
            ],
            'validation': 'Automated verification',
            'classification': 'ML-based categorization'
        },

        'initial_response': {
            'acknowledgment': 'Auto-generated',
            'assignment': 'Skills-based routing',
            'notification': 'Stakeholder alerts',
            'war_room': 'Virtual space creation'
        },

        'investigation': {
            'data collection': [
                'System logs',
                'Performance metrics',
                'Configuration state',
                'Recent changes'
            ],
            'root_cause_analysis': {
                'method': '5-whys + fishbone',
                'tools': 'AI-assisted RCA',
                'documentation': 'Real-time wiki'
            }
        },

        'resolution': {
            'fix_implementation': 'Change advisory board',
            'testing': 'Automated validation',
            'deployment': 'Canary rollout',
            'verification': 'Customer confirmation'
        },

        'closure': {
            'documentation': 'Incident report',
            'lessons learned': 'Blameless postmortem',
            'knowledge base': 'Article creation',
            'prevention': 'Proactive measures'
        }
    }

    return workflow
```

---

3. TROUBLESHOOTING GUIDES

3.1 Common Issues and Resolutions

```
troubleshooting_database:
  quantum_gateway_issues:
    symptom: "Agents cannot connect to quantum gateway"
    diagnostic_steps:
      1: "Check gateway status: curl https://quantum.mwrasp.ai/health"
      2: "Verify quantum certificates: openssl x509 -in cert.pem -
text"
      3: "Test quantum handshake: mwrasp-cli test-quantum"
      4: "Review gateway logs: tail -f /var/log/mwrasp/quantum-
gateway.log"

    common_causes:
      - Certificate expiration
      - Firewall blocking port 9443
      - Quantum entropy depletion
      - Network MTU issues

    resolution_steps:
      certificate_renewal: |
        mwrasp-cli cert renew --type quantum
        systemctl restart mwrasp-gateway

      firewall_configuration: |
        firewall-cmd --add-port=9443/tcp --permanent
        firewall-cmd --reload

      entropy_restoration: |
        systemctl restart rng-tools
        cat /proc/sys/kernel/random/entropy_avail

      mtu_adjustment: |
        ip link set dev eth0 mtu 1450
        echo "net.ipv4.tcp_mtu_probing=1" >> /etc/sysctl.conf

  ai_agent_behavioral_issues:
    symptom: "Agents showing anomalous behavior patterns"
    diagnostic_steps:
      1: "Review behavioral baseline: mwrasp-cli agent baseline --id
AGENT_ID"
      2: "Check drift metrics: mwrasp-cli agent drift-analysis"
      3: "Analyze recent updates: git log --oneline -10"
      4: "Verify model version: mwrasp-cli model status"

    resolution_matrix:
      behavioral_recalibration: |
        mwrasp-cli agent recalibrate --id AGENT_ID --baseline last-
known-good
        mwrasp-cli agent monitor --real-time

      model_rollback: |
        mwrasp-cli model rollback --version previous
```

```
        mwrasp-cli agent restart --all

    quarantine_procedure: |
        mwrasp-cli agent quarantine --id AGENT_ID
        mwrasp-cli agent analyze --deep --id AGENT_ID
        mwrasp-cli agent restore --clean --id AGENT_ID
```

3.2 Performance Optimization

```
class PerformanceOptimization:
    def optimization_playbook(self):
        return {
            'latency_reduction': {
                'diagnosis': {
                    'tools': ['tcpdump', 'wireshark', 'mwrasp-perf'],
                    'metrics': ['RTT', 'Processing time', 'Queue
depth'],
                    'thresholds': {
                        'acceptable': '<100ms',
                        'warning': '100-500ms',
                        'critical': '>500ms'
                    }
                },

                'optimizations': {
                    'network': [
                        'Enable TCP fast open',
                        'Optimize TCP window scaling',
                        'Implement connection pooling',
                        'Deploy edge gateways'
                    ],

                    'processing': [
                        'Enable caching layers',
                        'Implement batch processing',
                        'Optimize database queries',
                        'Use quantum acceleration'
                    ],

                    'configuration': {
                        'worker threads': 'CPU cores * 2',
                        'connection pool': '100-500 connections',
                        'cache size': '10GB minimum',
                        'quantum_threads': '8-16 optimal'
                    }
                }
            },

            'throughput_enhancement': {
```

```
                'bottleneck_analysis': {
                    'cpu': 'perf top -p PID',
                    'memory': 'pmap -x PID',
                    'disk': 'iotop -p PID',
                    'network': 'iftop -i INTERFACE'
                },

                'scaling_strategies': {
                    'horizontal': {
                        'auto_scaling': 'CPU > 70% for 5 minutes',
                        'load_balancing': 'Least connections',
                        'session_affinity': 'IP hash'
                    },

                    'vertical': {
                        'cpu': '16 cores minimum',
                        'memory': '64GB recommended',
                        'disk': 'NVMe SSD required',
                        'network': '10Gbps preferred'
                    }
                }
            }
        }
```

---

4. MONITORING AND ALERTING

4.1 System Health Monitoring

```
class SystemHealthMonitoring:
    def monitoring_framework(self):
        return {
            'infrastructure monitoring': {
                'metrics': {
                    'cpu utilization': {
                        'collection': 'Every 10 seconds',
                        'threshold warning': '70%',
                        'threshold critical': '90%',
                        'action': 'Auto-scale or alert'
                    },

                    'memory usage': {
                        'collection': 'Every 10 seconds',
                        'threshold warning': '80%',
                        'threshold critical': '95%',
                        'action': 'Memory dump and restart'
                    },
```

```
                    'disk_usage': {
                        'collection': 'Every minute',
                        'threshold_warning': '75%',
                        'threshold_critical': '90%',
                        'action': 'Log rotation and cleanup'
                    },

                    'network_throughput': {
                        'collection': 'Every 10 seconds',
                        'threshold_warning': '80% capacity',
                        'threshold_critical': '95% capacity',
                        'action': 'Traffic shaping'
                    }
                },

                'tools': {
                    'prometheus': {
                        'scrape_interval': '15s',
                        'retention': '15 days',
                        'storage': 'Time-series database'
                    },

                    'grafana': {
                        'dashboards': 47,
                        'alerts': 234,
                        'integrations': ['Slack', 'PagerDuty',
'Email']
                    }
                }
            },

            'application_monitoring': {
                'apm_metrics': {
                    'response_time': 'p50, p95, p99',
                    'error_rate': 'Errors per minute',
                    'throughput': 'Requests per second',
                    'saturation': 'Queue depth'
                },

                'distributed_tracing': {
                    'tool': 'Jaeger',
                    'sampling_rate': '1%',
                    'retention': '7 days',
                    'correlation': 'TraceID propagation'
                },

                'log_aggregation': {
                    'tool': 'ELK Stack',
                    'retention': '30 days hot, 90 days warm',
                    'indexing': 'Daily indices',
                    'search': 'Full-text with ML'
                }
```

```
            },

            'quantum_specific_monitoring': {
                'qubit_health': {
                    'coherence_time': 'Continuous measurement',
                    'gate_fidelity': 'Per operation tracking',
                    'error_rates': 'Real-time calculation'
                },

                'entropy_levels': {
                    'pool_size': 'Minimum 4096 bits',
                    'generation_rate': '>1Mbps',
                    'quality_metrics': 'NIST SP 800-90B'
                },

                'algorithm_performance': {
                    'shor_factorization': 'Success rate tracking',
                    'grover_search': 'Speedup measurement',
                    'quantum_supremacy': 'Benchmark comparison'
                }
            }
        }
```

4.2 Alert Configuration

```
alert_configuration:
 alert_rules:
   critical_alerts:
     quantum_gateway_down:
       condition: "up{job='quantum-gateway'} == 0"
       duration: "1 minute"
       action:
         - Page on-call engineer
         - Create P1 incident
         - Initiate failover

     security_breach_detected:
       condition: "security_score < 70 OR intrusion_detected == 1"
       duration: "Immediate"
       action:
         - Page security_team
         - Isolate affected_systems
         - Preserve forensic_data

     data_corruption:
       condition: "data_integrity_check == 'failed'"
       duration: "Immediate"
       action:
         - Stop_writes
```

```
        - Initiate recovery
        - Notify data team

    warning_alerts:
      high_latency:
        condition: "response_time_p95 > 500ms"
        duration: "5 minutes"
        action:
          - Notify operations team
          - Scale resources
          - Analyze bottlenecks

      certificate_expiry:
        condition: "days_until_expiry < 30"
        duration: "Daily check"
        action:
          - Email administrators
          - Create renewal ticket
          - Schedule maintenance

    info_alerts:
      successful_deployment:
        condition: "deployment_status == 'success'"
        action:
          - Update status page
          - Notify stakeholders
          - Log in audit trail

      scheduled_maintenance:
        condition: "maintenance_window == 'active'"
        action:
          - Display banner
          - Send reminders
          - Track progress

  notification_channels:
    pagerduty:
      integration_key: "${PAGERDUTY_KEY}"
      escalation_policy: "Follow-the-sun"
      deduplication: "incident_key"

    slack:
      webhook_url: "${SLACK_WEBHOOK}"
      channels:
        critical: "#incidents-p1"
        warning: "#alerts-warning"
        info: "#general-notifications"

    email:
      smtp_server: "smtp.mwrasp.ai"
      from_address: "alerts@mwrasp.ai"
      distribution_lists:
```

```
        critical: "oncall@mwrasp.ai"
        warning: "ops@mwrasp.ai"
        info: "team@mwrasp.ai"
```

---

5. KNOWLEDGE BASE MANAGEMENT

5.1 Documentation Structure

```
class KnowledgeBaseSystem:
    def documentation_hierarchy(self):
        return {
            'customer_facing': {
                'getting_started': {
                    'quick_start_guide': 'pages/12',
                    'installation_guide': 'pages/34',
                    'configuration_basics': 'pages/28',
                    'first_deployment': 'pages/18'
                },

                'user_guides': {
                    'admin_guide': 'pages/156',
                    'operator_guide': 'pages/98',
                    'developer_guide': 'pages/234',
                    'security_guide': 'pages/187'
                },

                'troubleshooting': {
                    'common_issues': 'articles/847',
                    'error_codes': 'articles/423',
                    'faqs': 'articles/234',
                    'best_practices': 'articles/156'
                },

                'api_documentation': {
                    'rest_api': 'OpenAPI 3.0 spec',
                    'graphql_api': 'Schema documentation',
                    'websocket_api': 'Event specifications',
                    'sdk_references': ['Python', 'Go', 'Java',
'Node.js']
                }
            },

            'internal_documentation': {
                'runbooks': {
                    'incident_response': 67,
                    'maintenance_procedures': 45,
                    'deployment_guides': 34,
```

```
                    'rollback_procedures': 23
                },

                'architecture': {
                    'system_design': 'Detailed diagrams',
                    'data_flow': 'Sequence diagrams',
                    'security_architecture': 'Threat models',
                    'quantum_algorithms': 'Implementation details'
                },

                'training_materials': {
                    'onboarding': '5-day program',
                    'certification_prep': 'Study guides',
                    'skill_development': 'Learning paths',
                    'lab_exercises': 'Hands-on scenarios'
                }
            }
        }

    def search_optimization(self):
        return {
            'indexing': {
                'full_text': 'Elasticsearch',
                'metadata': 'Tagged categorization',
                'relevance': 'ML-powered ranking',
                'suggestions': 'Auto-complete enabled'
            },

            'analytics': {
                'popular_articles': 'View tracking',
                'search_queries': 'Gap analysis',
                'user_feedback': 'Rating system',
                'improvement_metrics': 'Monthly review'
            }
        }
```

5.2 Self-Service Portal

```
self service_portal:
 features:
   ticket management:
     submission:
       - Web form with guided workflow
       - Email integration
       - API submission
       - Chat bot interface

     tracking:
       - Real-time status updates
```

```
        - SLA countdown
        - Escalation visibility
        - Resolution history

      automation:
        - Auto-categorization
        - Suggested solutions
        - Similar ticket matching
        - Predictive routing

    knowledge_search:
      capabilities:
        - Natural language processing
        - Contextual recommendations
        - Video tutorials
        - Interactive diagrams

      personalization:
        - Role-based content
        - History tracking
        - Bookmarks and favorites
        - Custom collections

    community_forum:
      sections:
        - General discussions
        - Feature requests
        - Best practices
        - Success stories

      gamification:
        - Reputation points
        - Expert badges
        - Solution marking
        - Leaderboards

    service_catalog:
      request_types:
        - Account management
        - Access requests
        - Configuration changes
        - Training enrollment

      approval_workflow:
        - Manager approval
        - Security review
        - Automated provisioning
        - Completion notification
```

---

6. REMOTE SUPPORT CAPABILITIES

6.1 Remote Diagnostic Tools

```
class RemoteDiagnostics:
    def diagnostic_toolkit(self):
        return {
            'secure remote access': {
                'protocol': 'Quantum-secure SSH',
                'authentication': {
                    'method': 'Certificate + TOTP',
                    'approval': 'Customer consent required',
                    'session recording': 'Mandatory',
                    'time_limit': '2 hours maximum'
                },

                'capabilities': [
                    'Read-only log access',
                    'Performance monitoring',
                    'Configuration review',
                    'Diagnostic script execution'
                ],

                'restrictions': [
                    'No production data access',
                    'No configuration changes without approval',
                    'Audit trail mandatory',
                    'Break-glass procedures logged'
                ]
            },

            'automated diagnostics': {
                'health checks': {
                    'system vitals': self.check system health(),
                    'connectivity test': self.test_connectivity(),
                    'configuration_validation':
self.validate config(),
                    'security_posture': self.assess_security()
                },

                'data collection': {
                    'log_bundle': 'Automated collection and
sanitization',
                    'metrics snapshot': 'Point-in-time capture',
                    'configuration dump': 'Redacted sensitive data',
                    'trace_collection': 'Performance profiling'
                },

                'analysis_tools': {
```

```
                    'log_analysis': 'ML-powered pattern recognition',
                    'anomaly_detection': 'Behavioral analysis',
                    'root_cause_analysis': 'Automated RCA engine',
                    'recommendation_engine': 'Fix suggestions'
                }
            }
        }

    def remote fix_capabilities(self):
        return {
            'automated_remediation': {
                'safe_fixes': [
                    'Service restarts',
                    'Cache clearing',
                    'Certificate renewal',
                    'Connection reset'
                ],

                'approval_required': [
                    'Configuration changes',
                    'Software updates',
                    'Security modifications',
                    'Data operations'
                ],

                'rollback_capability': 'All changes reversible',
                'testing_required': 'Automated validation'
            }
        }
```

6.2 Support Analytics Dashboard

```
def support_analytics_dashboard():
    """
    Real-time support metrics and analytics
    """
    return {
        'operational_metrics': {
            'ticket_volume': {
                'daily_average': 347,
                'peak_hour': '10 AM EST',
                'trend': '+12% MoM',
                'forecast': 'ML-based prediction'
            },

            'resolution_metrics': {
                'first_contact_resolution': '67%',
                'average_resolution_time': '4.7 hours',
                'escalation_rate': '33%',
```

```
                'reopen_rate': '3.2%'
            },

            'sla_compliance': {
                'p1_compliance': '99.8%',
                'p2_compliance': '98.7%',
                'p3_compliance': '97.2%',
                'p4_compliance': '99.1%'
            }
        },

        'quality_metrics': {
            'customer_satisfaction': {
                'csat_score': '98.7%',
                'nps_score': 72,
                'ces_score': 4.8,
                'survey_response_rate': '42%'
            },

            'agent_performance': {
                'average_handle_time': '8.3 minutes',
                'quality_score': '94%',
                'training_completion': '100%',
                'certification_rate': '87%'
            }
        },

        'predictive_analytics': {
            'issue_prediction': {
                'accuracy': '84%',
                'lead_time': '4 hours average',
                'prevention_rate': '62%',
                'false_positive_rate': '8%'
            },

            'capacity_planning': {
                'staffing_optimization': 'ML-based scheduling',
                'skill_gap_analysis': 'Quarterly assessment',
                'training_recommendations': 'Personalized paths',
                'workload_balancing': 'Real-time adjustment'
            }
        }
    }
```

---

7. TRAINING AND CERTIFICATION

7.1 Support Team Training Program

```
training_program:
 onboarding:
   week_1:
     topics:
       - Company overview and culture
       - Product fundamentals
       - Quantum computing basics
       - AI agent architecture

     activities:
       - Interactive workshops
       - Lab exercises
       - Shadow experienced agents
       - Knowledge base exploration

   week_2:
     topics:
       - Support tools and systems
       - Ticketing system mastery
       - Communication skills
       - Customer empathy training

     activities:
       - Mock support calls
       - Ticket handling practice
       - Role-playing exercises
       - Tool certifications

   week 3:
     topics:
       - Technical deep dives
       - Troubleshooting methodology
       - Security procedures
       - Compliance requirements

     activities:
       - Hands-on troubleshooting
       - Security incident drills
       - Compliance training
       - Technical assessments

   week 4:
     topics:
       - Advanced diagnostics
       - Escalation procedures
       - Quality standards
       - Performance metrics

     activities:
       - Live ticket handling (supervised)
       - Quality reviews
```

```
        - Feedback sessions
        - Final certification exam

  ongoing_training:
    technical_skills:
      frequency: "Weekly tech talks"
      topics:
        - New feature training
        - Quantum algorithm updates
        - Security threat briefings
        - Performance optimization

    soft_skills:
      frequency: "Monthly workshops"
      topics:
        - Communication excellence
        - Conflict resolution
        - Cultural sensitivity
        - Stress management

    certifications:
      internal:
        - MWRASP Certified Support Engineer
        - Quantum Systems Specialist
        - AI Security Expert
        - Senior Troubleshooting Master

      external:
        - CompTIA Security+
        - ITIL Foundation
        - AWS/Azure/GCP certifications
        - Quantum computing certificates
```

7.2 Customer Training Resources

```
class CustomerTraining:
    def training_catalog(self):
        return {
            'self_paced_learning': {
                'online_courses': {
                    'mwrasp_fundamentals': {
                        'duration': '8 hours',
                        'modules': 12,
                        'certification': 'Yes',
                        'validity': '2 years'
                    },

                    'advanced_administration': {
                        'duration': '16 hours',
```

```
                        'modules': 24,
                        'prerequisites': 'Fundamentals',
                        'certification': 'Yes'
                    },

                    'quantum_security_mastery': {
                        'duration': '40 hours',
                        'modules': 45,
                        'prerequisites': 'Advanced Admin',
                        'certification': 'Expert level'
                    }
                },

                'learning_paths': {
                    'administrator': ['Fundamentals', 'Admin',
'Security'],
                    'developer': ['Fundamentals', 'API',
'Integration'],
                    'security_analyst': ['Fundamentals', 'Security',
'Quantum'],
                    'architect': ['All courses recommended']
                }
            },

            'instructor_led_training': {
                'virtual_classes': {
                    'frequency': 'Weekly',
                    'duration': '4 hours',
                    'capacity': '20 students',
                    'recording': 'Available for 30 days'
                },

                'on_site_training': {
                    'availability': 'Customer location',
                    'duration': '3-5 days',
                    'customization': 'Tailored content',
                    'hands_on_labs': 'Included'
                }
            },

            'certification_program': {
                'levels': {
                    'associate': {
                        'experience': '6 months',
                        'exam': '90 minutes, 60 questions',
                        'passing_score': '70%',
                        'renewal': '3 years'
                    },

                    'professional': {
                        'experience': '2 years',
                        'exam': '120 minutes, 80 questions',
```

```
                    'passing_score': '75%',
                    'renewal': '3 years'
                },

                'expert': {
                    'experience': '5 years',
                    'exam': '180 minutes, practical + theory',
                    'passing_score': '80%',
                    'renewal': '3 years'
                }
            },

            'benefits': [
                'Industry recognition',
                'Priority support',
                'Community access',
                'Conference discounts'
            ]
        }
    }
```

---

8. ESCALATION PROCEDURES

8.1 Escalation Matrix

```
class EscalationProcedures:
    def escalation_matrix(self):
        return {
            'technical escalation': {
                'tier1 to tier2': {
                    'criteria': [
                        'Issue beyond KB scope',
                        'Customer dissatisfaction',
                        'Time limit exceeded',
                        'Multiple failed attempts'
                    ],
                    'process': 'Warm transfer with context',
                    'sla': '15 minutes handoff'
                },

                'tier2 to tier3': {
                    'criteria': [
                        'Code-level investigation needed',
                        'Performance degradation >20%',
                        'Security incident suspected',
                        'Quantum algorithm issues'
                    ],
```

```
                    'process': 'Detailed ticket with diagnostics',
                    'sla': '30 minutes response'
                },

                'tier3_to_development': {
                    'criteria': [
                        'Product bug confirmed',
                        'Feature limitation',
                        'Patch required',
                        'Architecture change needed'
                    ],
                    'process': 'JIRA ticket with reproduction steps',
                    'sla': '4 hours for P1'
                }
            },

            'management escalation': {
                'triggers': [
                    'Customer executive complaint',
                    'SLA breach imminent',
                    'Revenue at risk',
                    'Legal implications'
                ],

                'chain_of_command': {
                    'level 1': 'Team Lead',
                    'level_2': 'Support Manager',
                    'level_3': 'Director of Support',
                    'level 4': 'VP Customer Success',
                    'level_5': 'CEO'
                },

                'response protocol': {
                    'acknowledgment': '<30 minutes',
                    'action plan': '<2 hours',
                    'executive briefing': '<4 hours',
                    'resolution_commitment': 'Within same day'
                }
            }
        }

    def escalation_communication(self):
        return {
            'internal communication': {
                'slack channels': {
                    'P1': '#escalation-p1',
                    'P2': '#escalation-p2',
                    'management': '#escalation-mgmt'
                },

                'war room protocol': {
                    'initiation': 'P1 or multiple P2s',
```

```
                    'participants': 'Cross-functional team',
                    'updates': 'Every 15 minutes',
                    'documentation': 'Real-time wiki'
                }
            },

            'customer communication': {
                'templates': {
                    'initial_escalation': 'Acknowledge and set
expectations',
                    'progress_update': 'Technical status and next
steps',
                    'resolution': 'Root cause and prevention',
                    'follow_up': '48-hour satisfaction check'
                },

                'communication cadence': {
                    'P1': 'Every 30 minutes',
                    'P2': 'Every 2 hours',
                    'P3': 'Every 8 hours',
                    'P4': 'Daily updates'
                }
            }
        }
```

---

9. INTEGRATION SUPPORT

9.1 Third-Party Integration Support

```yaml
integration_support:
 supported_platforms:
   siem_systems:
     splunk:
       version: "8.x, 9.x"
       integration_type: "Native app + API"
       documentation: "link/to/splunk/guide"
       support_level: "Full"

     qradar:
       version: "7.4+"
       integration_type: "DSM + REST API"
       documentation: "link/to/qradar/guide"
       support_level: "Full"

     elastic:
       version: "7.x, 8.x"
       integration_type: "Beats + Logstash"
```

```
        documentation: "link/to/elastic/guide"
        support_level: "Full"

    identity_providers:
      active_directory:
        protocols: ["LDAP", "SAML 2.0", "OAuth 2.0"]
        mfa_support: "Yes"
        group_sync: "Real-time"
        documentation: "link/to/ad/guide"

      okta:
        protocols: ["SAML 2.0", "OIDC"]
        provisioning: "SCIM 2.0"
        mfa_support: "Native"
        documentation: "link/to/okta/guide"

      azure_ad:
        protocols: ["SAML 2.0", "OAuth 2.0", "OIDC"]
        conditional_access: "Supported"
        group_sync: "Scheduled"
        documentation: "link/to/azure/guide"

    orchestration:
      kubernetes:
        deployment: "Helm charts provided"
        operators: "CRD-based"
        monitoring: "Prometheus metrics"
        scaling: "HPA/VPA supported"

      terraform:
        providers: "AWS, Azure, GCP"
        modules: "Pre-built available"
        state_management: "Remote backend"
        documentation: "IaC best practices"

  integration_testing:
    validation_suite:
      - Connectivity verification
      - Authentication testing
      - Data flow validation
      - Performance benchmarking
      - Security assessment

    certification_process:
      requirements:
        - Functional testing
        - Load testing
        - Security review
        - Documentation review
```

```
      timeline: "2-week process"
      support: "Dedicated integration engineer"
```

9.2 API Support

```
class APISupport:
   def api_support_framework(self):
       return {
           'api_documentation': {
               'formats': {
                   'rest_api': {
                       'specification': 'OpenAPI 3.0',
                       'interactive_docs': 'Swagger UI',
                       'postman_collection': 'Available',
                       'code_examples': ['Python', 'Go', 'Java',
'Node.js']
                   },

                   'graphql_api': {
                       'schema': 'SDL format',
                       'playground': 'GraphiQL',
                       'subscriptions': 'WebSocket',
                       'federation': 'Apollo compliant'
                   },

                   'grpc_api': {
                       'proto_files': 'Version controlled',
                       'client_libraries': 'Auto-generated',
                       'streaming': 'Bidirectional',
                       'load_balancing': 'Client-side'
                   }
               },

               'versioning': {
                   'strategy': 'Semantic versioning',
                   'deprecation_notice': '6 months',
                   'backward_compatibility': '2 major versions',
                   'migration_guides': 'Provided'
               }
           },

           'developer_support': {
               'sdks': {
                   'languages': ['Python', 'Go', 'Java',
'JavaScript', 'C#'],
                   'package_managers': ['pip', 'npm', 'maven',
'nuget'],
                   'documentation': 'Language-specific guides',
                   'examples': 'GitHub repository'
```

```
                },

                'testing_tools': {
                    'sandbox environment': 'sandbox.mwrasp.ai',
                    'test_data': 'Synthetic datasets',
                    'rate_limits': 'Relaxed for testing',
                    'reset_capability': 'Daily refresh'
                },

                'support_channels': {
                    'developer_forum': '24/7 community',
                    'slack channel': 'Direct access to engineers',
                    'office_hours': 'Weekly video calls',
                    'priority_support': 'Enterprise customers'
                }
            }
        }
```

---

10. CONTINUOUS IMPROVEMENT

10.1 Quality Assurance Program

```
class QualityAssurance:
    def qa_framework(self):
        return {
            'ticket quality review': {
                'sampling': {
                    'rate': '10% random sample',
                    'focus_areas': ['P1 incidents', 'Escalations',
'Low CSAT'],
                    'frequency': 'Daily reviews'
                },

                'evaluation criteria': {
                    'technical accuracy': 'Weight: 40%',
                    'communication quality': 'Weight: 30%',
                    'process adherence': 'Weight: 20%',
                    'customer_satisfaction': 'Weight: 10%'
                },

                'feedback loop': {
                    'individual coaching': 'Weekly 1:1s',
                    'team calibration': 'Monthly sessions',
                    'best practice sharing': 'Team meetings',
                    'recognition_program': 'Monthly awards'
                }
            },
```

```
            'process improvement': {
                'kaizen_events': {
                    'frequency': 'Quarterly',
                    'participants': 'Cross-functional',
                    'focus': 'Pain points and inefficiencies',
                    'outcomes': 'Action plans with owners'
                },

                'automation_opportunities': {
                    'identification': 'ML-based analysis',
                    'prioritization': 'ROI calculation',
                    'implementation': 'Agile sprints',
                    'measurement': 'Before/after metrics'
                },

                'knowledge_gap_analysis': {
                    'search_analytics': 'No-result queries',
                    'escalation_patterns': 'Repeat issues',
                    'customer_feedback': 'Survey insights',
                    'content_creation': 'Weekly KB updates'
                }
            }
        }

    def customer_feedback_management(self):
        return {
            'collection_methods': {
                'post_interaction_survey': 'CSAT, CES',
                'quarterly_nps': 'Relationship metric',
                'user_interviews': 'Monthly sessions',
                'advisory_board': 'Quarterly meetings'
            },

            'analysis_and_action': {
                'sentiment_analysis': 'NLP processing',
                'trend_identification': 'Statistical analysis',
                'root_cause_analysis': 'For detractors',
                'action_planning': 'SMART goals'
            },

            'closing_the_loop': {
                'individual_follow_up': 'For low scores',
                'feature_requests': 'Product roadmap input',
                'process_changes': 'Based on feedback',
                'communication': 'What we heard/did'
            }
        }
```

APPENDICES

A. Support Contact Information

```
support_contacts:
 customer_facing:
   phone:
     us: "+1-800-MWRASP1"
     eu: "+44-20-MWRASP1"
     apac: "+65-MWRASP100"

   email:
     general: "support@mwrasp.ai"
     critical: "critical@mwrasp.ai"
     escalation: "escalation@mwrasp.ai"

   portal: "https://support.mwrasp.ai"

   chat: "Available 24/7 on portal"

 internal_escalation:
   tier2_lead:
     name: "John Smith"
     phone: "+1-555-0111"
     email: "jsmith@mwrasp.ai"

   tier3_lead:
     name: "Maria Garcia"
     phone: "+1-555-0112"
     email: "mgarcia@mwrasp.ai"

   on_call_rotation:
     schedule: "PagerDuty"
     escalation: "15-minute response"
```

B. Common Error Codes

```
ERROR_CODES = {
   'QT-001': 'Quantum gateway timeout',
   'QT-002': 'Quantum certificate invalid',
   'AI-001': 'Agent behavioral anomaly detected',
   'AI-002': 'Model version mismatch',
   'SEC-001': 'Authentication failure',
   'SEC-002': 'Authorization denied',
   'NET-001': 'Network connectivity lost',
   'NET-002': 'API rate limit exceeded',
```

```
    'SYS-001': 'System resource exhaustion',
    'SYS-002': 'Service dependency failure'
}
```

C. Useful Commands Reference

```
 # System diagnostics
mwrasp-cli system status
mwrasp-cli system health --detailed
mwrasp-cli system diagnose --full

# Agent management
mwrasp-cli agent list --all
mwrasp-cli agent status --id AGENT_ID
mwrasp-cli agent restart --id AGENT_ID

# Quantum operations
mwrasp-cli quantum status
mwrasp-cli quantum test-connectivity
mwrasp-cli quantum rotate-keys

# Log analysis
mwrasp-cli logs tail --follow
mwrasp-cli logs search --pattern "ERROR"
mwrasp-cli logs bundle --case CASE_ID
```

Document Classification: Internal Use Last Updated: August 2025 Next Review: November 2025 Approval: VP Customer Success

---