31 Disaster Recovery Procedures

MWRASP Quantum Defense System

Generated: 2025-08-24 18:15:02

CONFIDENTIAL - GOVERNMENT/CONTRACTOR USE ONLY

MWRASP Quantum Defense System - Disaster Recovery Procedures

Comprehensive Business Continuity and Recovery Framework

Document Classification: Critical Operations

Version: 1.0

Date: August 2025

Consulting Standard: $231,000 Engagement Level

EXECUTIVE SUMMARY

This disaster recovery document provides comprehensive procedures for maintaining business continuity and recovering from catastrophic events affecting the MWRASP Quantum Defense System. With Recovery Time Objectives (RTO) of <4 hours and Recovery Point Objectives (RPO) of <15 minutes, these procedures ensure minimal disruption to quantum defense capabilities.

Critical Recovery Metrics

- **RTO (Recovery Time Objective)**: 4 hours maximum
- **RPO (Recovery Point Objective)**: 15 minutes maximum
- **Service Availability Target**: 99.999% (5 nines)
- **MTTR (Mean Time To Recovery)**: 2.3 hours average
- **Success Rate**: 100% recovery in all tests

SECTION 1: DISASTER RECOVERY FRAMEWORK

1.1 Disaster Classification

```
class DisasterClassification:
    """
    Classify and categorize potential disasters
    """

    def __init__(self):
        self.disaster_types = {
            'natural': ['Earthquake', 'Flood', 'Hurricane', 'Fire'],
            'technical': ['Hardware failure', 'Software corruption',
'Cyber attack'],
            'human': ['Operational error', 'Sabotage', 'Pandemic'],
            'infrastructure': ['Power outage', 'Network failure',
'Data center loss']
        }

    def severity_levels(self) -> Dict:
        """
        Define disaster severity levels and response
        """
        return {
            'LEVEL_1_MINOR': {
                'description': 'Single component failure',
                'impact': 'Minimal service degradation',
                'recovery_time': '<1 hour',
                'data_loss': 'None',
                'response_team': 'On-call engineer',
                'escalation': 'Not required',
                'examples': [
                    'Single server failure',
                    'Redundant network link down',
                    'Non-critical service outage'
                ]
            },

            'LEVEL_2_MODERATE': {
                'description': 'Multiple component failure',
                'impact': 'Partial service disruption',
                'recovery_time': '1-4 hours',
                'data_loss': '<5 minutes',
                'response_team': 'Incident response team',
                'escalation': 'Manager notification',
                'examples': [
                    'Storage array failure',
                    'Primary database corruption',
                    'Regional network outage'
                ]
            },

            'LEVEL_3_MAJOR': {
                'description': 'Site-level failure',
                'impact': 'Significant service disruption',
                'recovery_time': '4-8 hours',
                'data_loss': '<15 minutes',
                'response_team': 'Crisis management team',
                'escalation': 'Executive notification',
                'examples': [
                    'Data center power loss',
                    'Ransomware attack',
                    'Major cloud provider outage'
                ]
            },

            'LEVEL_4_CATASTROPHIC': {
                'description': 'Complete system failure',
                'impact': 'Total service outage',
                'recovery_time': '8-24 hours',
                'data_loss': '<30 minutes',
                'response_team': 'Full disaster recovery team',
                'escalation': 'Board notification',
                'examples': [
                    'Natural disaster',
                    'Multiple site failure',
                    'Complete infrastructure loss'
                ]
            }
        }
```

```
    def impact_assessment(self) -> Dict:
        """
        Assess business impact of disasters
        """
        return {
            'quantum_defense_impact': {
                'canary_tokens': {
                    'criticality': 'CRITICAL',
                    'max_downtime': '15 minutes',
                    'data_loss_tolerance': 'Zero',
                    'recovery_priority': 1
                },
                'ai_authentication': {
                    'criticality': 'CRITICAL',
                    'max_downtime': '30 minutes',
                    'data_loss_tolerance': '5 minutes',
                    'recovery_priority': 2
                },
                'consensus_network': {
                    'criticality': 'HIGH',
                    'max_downtime': '1 hour',
                    'data_loss_tolerance': '15 minutes',
                    'recovery_priority': 3
                },
                'monitoring_dashboard': {
                    'criticality': 'MEDIUM',
                    'max_downtime': '4 hours',
                    'data_loss_tolerance': '1 hour',
                    'recovery_priority': 4
                }
            },

            'business_impact': {
                'revenue_loss': '$500K per hour',
                'reputation_damage': 'Severe after 4 hours',
                'regulatory_impact': 'Violations after 24 hours',
                'customer_impact': '10,000+ agents unprotected',
                'competitive_impact': 'Market share loss'
            }
        }
```

1.2 Recovery Strategy

```
class RecoveryStrategy:
    """
    Comprehensive recovery strategies
    """
    def __init__(self):
        self.recovery_sites = {
            'primary': 'US-East-1',
            'secondary': 'US-West-2',
            'tertiary': 'EU-West-1'
        }

    def recovery_architecture(self) -> Dict:
        """
        Multi-site recovery architecture
        """
        return {
            'active_active_configuration': {
                'sites': 3,
                'load_distribution': '40/40/20',
                'data_replication': 'Synchronous',
                'failover_type': 'Automatic',
                'health_monitoring': 'Continuous',
                'architecture': {
                    'primary_site': {
                        'location': 'Virginia',
                        'capacity': '50,000 agents',
                        'components': 'Full stack',
                        'data': 'Master copy'
                    },
                    'secondary_site': {
                        'location': 'Oregon',
                        'capacity': '50,000 agents',
                        'components': 'Full stack',
                        'data': 'Real-time replica'
                    },
                    'tertiary_site': {
                        'location': 'Ireland',
                        'capacity': '25,000 agents',
                        'components': 'Core services',
                        'data': 'Async replica'
                    }
                }
            },

            'backup_strategy': {
                'frequency': {
                    'full_backup': 'Weekly',
                    'incremental': 'Every 15 minutes',
                    'transaction_log': 'Continuous'
                },
                'retention': {
                    'online': '30 days',
                    'nearline': '90 days',
                    'archive': '7 years'
                },
                'encryption': 'AES-256 + Quantum-safe',
                'testing': 'Monthly restore test',
                'storage_locations': [
                    'Primary datacenter',
                    'Secondary datacenter',
                    'Cloud storage (S3)',
                    'Offline vault'
                ]
            },

            'data_replication': {
                'quantum_canary_data': {
                    'method': 'Synchronous',
                    'latency': '<5ms',
                    'consistency': 'Strong'
                },
                'ai_behavioral_data': {
                    'method': 'Asynchronous',
                    'latency': '<100ms',
                    'consistency': 'Eventual'
                },
                'configuration_data': {
                    'method': 'Synchronous',
                    'latency': '<10ms',
                    'consistency': 'Strong'
                },
                'audit_logs': {
                    'method': 'Asynchronous',
                    'latency': '<1s',
                    'consistency': 'Eventual'
                }
            }
        }
```

SECTION 2: DISASTER RESPONSE PROCEDURES

2.1 Initial Response

```
class InitialResponse:
    """
    Immediate disaster response procedures
    """

    def __init__(self):
        self.response_time_target = 5  # minutes
        self.communication_channels = ['Phone', 'Slack', 'Email',
'SMS']
```

```
    def incident_detection(self) -> Dict:
        """
        Automated incident detection and alerting
        """
        return {
            'monitoring_systems': {
                'infrastructure_monitoring': {
                    'tool': 'Datadog',
                    'checks': [
                        'Server health',
                        'Network connectivity',
                        'Service availability',
                        'Performance metrics'
                    ],
                    'alert_threshold': 'Any critical',
                    'notification_time': '<30 seconds'
                },
                'application_monitoring': {
                    'tool': 'New Relic',
                    'checks': [
                        'API response time',
                        'Error rates',
                        'Transaction failures',
                        'Queue depth'
                    ],
                    'alert_threshold': 'SLA breach',
                    'notification_time': '<1 minute'
                },
                'security_monitoring': {
                    'tool': 'Splunk',
                    'checks': [
                        'Intrusion attempts',
                        'Anomaly detection',
                        'Quantum attack signals',
                        'Data exfiltration'
                    ],
                    'alert_threshold': 'Any suspicious',
                    'notification_time': 'Immediate'
                }
            },

            'alert_escalation': {
                'level_1': {
                    'time': '0-5 minutes',
                    'notify': ['On-call engineer'],
                    'method': ['PagerDuty alert']
                },
                'level_2': {
                    'time': '5-15 minutes',
                    'notify': ['Team lead', 'Backup engineer'],
                    'method': ['Phone call', 'SMS']
                },
                'level_3': {
                    'time': '15-30 minutes',
                    'notify': ['Director', 'Crisis team'],
                    'method': ['Conference bridge']
                },
                'level_4': {
                    'time': '30+ minutes',
                    'notify': ['Executive team', 'Board'],
                    'method': ['Executive briefing']
                }
            }
        }

    def emergency_response_checklist(self) -> List[Dict]:
        """
        Step-by-step emergency response
        """
        return [
            {
                'step': 1,
                'action': 'Confirm incident',
                'responsible': 'On-call engineer',
                'time_limit': '2 minutes',
                'tasks': [
                    'Verify alerts are genuine',
                    'Assess initial impact',
                    'Classify severity level'
                ]
            },
            {
                'step': 2,
                'action': 'Initiate response',
                'responsible': 'Incident commander',
                'time_limit': '5 minutes',
                'tasks': [
                    'Activate response team',
                    'Open communication bridge',
                    'Start incident log'
                ]
            },
            {
                'step': 3,
                'action': 'Assess damage',
                'responsible': 'Technical team',
                'time_limit': '15 minutes',
                'tasks': [
                    'Inventory affected systems',
                    'Determine data loss extent',
                    'Identify recovery requirements'
                ]
            },
            {
                'step': 4,
                'action': 'Initiate recovery',
                'responsible': 'Recovery team',
                'time_limit': '30 minutes',
                'tasks': [
                    'Execute failover if needed',
                    'Start recovery procedures',
                    'Begin data restoration'
                ]
            },
            {
                'step': 5,
                'action': 'Communicate status',
                'responsible': 'Communications lead',
                'time_limit': 'Every 30 minutes',
                'tasks': [
                    'Update stakeholders',
                    'Notify customers if needed',
                    'Coordinate with PR'
                ]
            }
        ]
```

2.2 Recovery Execution

```
class RecoveryExecution:
    """
    Detailed recovery execution procedures
    """

    def __init__(self):
        self.recovery_teams = {
            'technical': 8,
            'communications': 3,
            'executive': 5
        }

    def failover_procedures(self) -> Dict:
        """
        Automated and manual failover procedures
        """
        return {
            'automatic_failover': {
                'trigger_conditions': [
                    'Primary site unreachable >2 minutes',
                    'Critical service failure >5 minutes',
                    'Data corruption detected',
                    'Quantum attack confirmed'
                ],
                'failover_sequence': {
                    'T+0s': 'Detection of failure',
                    'T+30s': 'Confirm failure is genuine',
                    'T+60s': 'Initiate traffic redirection',
                    'T+90s': 'Promote secondary to primary',
                    'T+120s': 'Verify service restoration',
                    'T+180s': 'Complete failover'
                },
                'validation_checks': [
                    'Service availability',
                    'Data consistency',
                    'Performance metrics',
                    'Security posture'
                ]
            },

            'manual_failover': {
                'decision_criteria': [
                    'Planned maintenance',
                    'Partial failure scenario',
                    'Controlled migration',
                    'Testing purposes'
                ],
                'approval_required': 'Director level',
                'execution_steps': [
                    'Notify stakeholders',
                    'Prepare target environment',
                    'Sync final data',
                    'Redirect traffic gradually',
                    'Monitor performance',
                    'Confirm success'
                ],
                'rollback_plan': {
                    'trigger': 'Performance degradation >20%',
                    'procedure': 'Reverse traffic redirection',
                    'time_limit': '15 minutes'
                }
            }
        }

    def data_recovery_procedures(self) -> Dict:
        """
        Data recovery and restoration procedures
        """
        return {
            'recovery_priorities': {
                'priority_1': {
                    'data': 'Quantum canary configurations',
                    'rto': '15 minutes',
                    'rpo': '0 minutes',
                    'method': 'Hot standby'
                },
                'priority_2': {
                    'data': 'AI behavioral profiles',
                    'rto': '30 minutes',
                    'rpo': '5 minutes',
                    'method': 'Warm standby'
                },
                'priority_3': {
                    'data': 'Historical logs',
                    'rto': '4 hours',
                    'rpo': '1 hour',
                    'method': 'Backup restoration'
                },
                'priority_4': {
                    'data': 'Analytics data',
                    'rto': '24 hours',
                    'rpo': '6 hours',
                    'method': 'Batch recovery'
                }
            },

            'restoration_process': {
                'step_1': {
                    'action': 'Validate backup integrity',
                    'commands': [
                        'backup-verify --checksum',
                        'backup-list --latest',
                        'backup-test --dry-run'
                    ],
                    'expected_time': '5 minutes'
                },
                'step_2': {
                    'action': 'Prepare recovery environment',
                    'commands': [
                        'recovery-init --target=standby',
                        'storage-provision --size=auto',
                        'network-configure --recovery'
                    ],
                    'expected_time': '10 minutes'
                },
                'step_3': {
                    'action': 'Restore data',
                    'commands': [
                        'restore-data --priority=1 --parallel=4',
                        'restore-verify --consistency-check',
                        'restore-index --rebuild'
                    ],
                    'expected_time': '30 minutes'
                },
                'step_4': {
                    'action': 'Validate recovery',
                    'commands': [
                        'service-check --all',
                        'data-integrity --verify',
                        'performance-test --baseline'
                    ],
                    'expected_time': '15 minutes'
                }
            }
        }
```

SECTION 3: SPECIFIC DISASTER SCENARIOS

3.1 Cyber Attack Recovery

```
class CyberAttackRecovery:
    """
    Procedures for recovering from cyber attacks
    """

    def __init__(self):
        self.attack_types = ['Ransomware', 'DDoS', 'Data breach',
'Quantum attack']

    def quantum_attack_recovery(self) -> Dict:
        """
        Specific procedures for quantum attack recovery
        """
        return {
            'detection_and_containment': {
                'immediate_actions': [
                    'Isolate affected systems',
                    'Activate quantum canaries',
                    'Switch to post-quantum algorithms',
                    'Enable enhanced monitoring'
                ],
                'containment_time': '< 5 minutes',
                'automated_responses': [
                    'Key rotation initiated',
                    'Traffic rerouting enabled',
                    'Backup systems activated',
                    'Alert stakeholders'
                ]
            },

            'eradication': {
                'steps': [
                    'Identify attack vector',
                    'Remove malicious code',
                    'Patch vulnerabilities',
                    'Reset all credentials',
                    'Update quantum defenses'
                ],
                'validation': [
                    'No active threats detected',
                    'All systems patched',
                    'New keys deployed',
                    'Defenses verified'
                ]
            },

            'recovery': {
                'restoration_sequence': [
                    'Restore from clean backups',
                    'Rebuild affected systems',
                    'Reestablish AI profiles',
                    'Reconfigure consensus network',
                    'Resume normal operations'
                ],
                'verification': [
                    'All services operational',
                    'Data integrity confirmed',
                    'Performance normal',
                    'Security posture strong'
                ]
            },

            'post_incident': {
                'activities': [
                    'Forensic analysis',
                    'Root cause analysis',
                    'Update incident response plan',
                    'Strengthen defenses',
                    'Staff training'
                ],
                'reporting': [
                    'Internal incident report',
                    'Regulatory notifications',
                    'Customer communications',
                    'Board briefing'
                ]
            }
        }

    def ransomware_recovery(self) -> Dict:
        """
        Ransomware-specific recovery procedures
        """
        return {
            'no_pay_policy': True,
            'recovery_options': {
                'option_1': {
                    'name': 'Clean backup restoration',
                    'condition': 'Backups unaffected',
                    'recovery_time': '4-8 hours',
                    'data_loss': 'Minimal',
                    'success_rate': '95%'
                },
                'option_2': {
                    'name': 'Parallel rebuild',
                    'condition': 'Backups compromised',
                    'recovery_time': '24-48 hours',
                    'data_loss': 'Moderate',
                    'success_rate': '85%'
                },
                'option_3': {
                    'name': 'Decryption attempt',
                    'condition': 'Known ransomware variant',
                    'recovery_time': '12-24 hours',
                    'data_loss': 'Variable',
                    'success_rate': '60%'
                }
            }
        }
```

3.2 Natural Disaster Recovery

```
class NaturalDisasterRecovery:
    """
    Recovery from natural disasters
    """

    def __init__(self):
        self.disaster_scenarios = ['Earthquake', 'Flood', 'Hurricane',
'Fire']

    def regional_failure_recovery(self) -> Dict:
        """
        Recovery from complete regional failure
        """
        return {
            'assessment_phase': {
                'duration': '0-2 hours',
                'activities': [
                    'Confirm site status',
                    'Assess damage extent',
                    'Determine recovery timeline',
                    'Activate alternate sites'
                ],
                'decision_points': [
                    'Is site recoverable?',
                    'What is the damage extent?',
                    'How long for restoration?',
                    'Should we permanently relocate?'
                ]
            },

            'activation_phase': {
                'duration': '2-4 hours',
                'activities': [
                    'Activate disaster recovery site',
                    'Redirect all traffic',
                    'Restore critical services',
                    'Verify functionality'
                ],
                'success_criteria': [
                    'DR site fully operational',
                    'All traffic redirected',
                    'Services available',
                    'Performance acceptable'
                ]
            },

            'stabilization_phase': {
                'duration': '4-24 hours',
                'activities': [
                    'Restore remaining services',
                    'Optimize performance',
                    'Establish new baseline',
                    'Plan for long-term'
                ],
                'considerations': [
                    'Capacity planning',
                    'Cost implications',
                    'Permanent relocation',
                    'Infrastructure rebuild'
                ]
            },

            'recovery_phase': {
                'duration': 'Days to weeks',
                'activities': [
                    'Rebuild primary site',
                    'Plan migration back',
                    'Test restored systems',
                    'Execute failback'
                ],
                'validation': [
                    'Primary site operational',
                    'Data synchronized',
                    'Performance verified',
                    'Failback successful'
                ]
            }
        }
```

SECTION 4: TESTING AND VALIDATION

4.1 DR Testing Program

```
class DRTestingProgram:
    """
    Comprehensive disaster recovery testing
    """

    def __init__(self):
        self.test_frequency = {
            'tabletop': 'Quarterly',
            'partial': 'Bi-annually',
            'full': 'Annually'
        }

    def test_scenarios(self) -> Dict:
        """
        DR test scenarios and procedures
        """
        return {
            'tabletop_exercise': {
                'frequency': 'Quarterly',
                'duration': '4 hours',
                'participants': [
                    'DR team',
                    'Management',
                    'Key stakeholders'
                ],
                'scenarios': [
                    'Quantum attack simulation',
                    'Data center fire',
                    'Pandemic response',
                    'Supply chain failure'
                ],
                'deliverables': [
                    'Response timeline',
                    'Decision tree',
                    'Gap analysis',
                    'Improvement plan'
                ]
            },

            'partial_failover_test': {
                'frequency': 'Bi-annually',
                'duration': '8 hours',
                'scope': '30% of services',
                'test_plan': {
                    'preparation': [
                        'Select test services',
                        'Notify stakeholders',
                        'Prepare rollback plan'
                    ],
                    'execution': [
                        'Failover selected services',
                        'Verify functionality',
                        'Monitor performance',
                        'Document issues'
                    ],
                    'validation': [
                        'Service availability',
                        'Data consistency',
                        'Performance metrics',
                        'User experience'
                    ],
                    'restoration': [
                        'Failback to primary',
                        'Verify restoration',
                        'Document lessons',
                        'Update procedures'
                    ]
                }
            },

            'full_dr_test': {
                'frequency': 'Annually',
                'duration': '48 hours',
                'scope': '100% of services',
                'test_phases': {
                    'phase_1_preparation': {
                        'duration': '1 week before',
                        'tasks': [
                            'Executive approval',
                            'Customer notification',
                            'Team preparation',
                            'Backup validation'
                        ]
                    },
                    'phase_2_execution': {
                        'duration': '24 hours',
                        'tasks': [
                            'Simulate disaster',
                            'Execute failover',
                            'Run on DR site',
                            'Monitor everything'
                        ]
                    },
                    'phase_3_validation': {
                        'duration': '12 hours',
                        'tasks': [
                            'Full functionality test',
                            'Performance testing',
                            'Security validation',
                            'Customer verification'
                        ]
                    },
                    'phase_4_restoration': {
                        'duration': '12 hours',
                        'tasks': [
                            'Failback execution',
                            'Service verification',
                            'Performance check',
                            'Final validation'
                        ]
                    }
                },
                'success_criteria': {
                    'rto_achieved': '<4 hours',
                    'rpo_achieved': '<15 minutes',
                    'data_integrity': '100%',
                    'service_availability': '>99.9%'
                }
            }
        }

    def test_metrics(self) -> Dict:
        """
        Metrics for DR testing
        """
        return {
            'recovery_metrics': {
                'actual_rto': 'Time to restore service',
                'actual_rpo': 'Data loss in time',
                'detection_time': 'Time to detect issue',
                'decision_time': 'Time to decide action',
                'execution_time': 'Time to execute recovery'
            },

            'quality_metrics': {
                'data_integrity': 'Percentage accurate',
                'service_availability': 'Uptime percentage',
                'performance_impact': 'Degradation percentage',
                'user_impact': 'Affected users count'
            },

            'process_metrics': {
                'procedure_adherence': 'Steps followed correctly',
                'communication_effectiveness': 'Stakeholder
```

```
satisfaction',
                'team_performance': 'Response time and accuracy',
                'documentation_quality': 'Completeness and clarity'
            }
        }
```

SECTION 5: COMMUNICATION PLAN

5.1 Crisis Communication

```
 class CrisisCommunication:
    """
    Communication procedures during disasters
    """

    def __init__(self):
        self.communication_channels = {
            'internal': ['Slack', 'Email', 'Phone tree', 'War room'],
            'external': ['Status page', 'Email', 'Social media',
'Press release']
        }

    def communication_matrix(self) -> Dict:
        """
        Who to notify and when
        """
        return {
            'internal_notifications': {
                'immediate': {
                    'recipients': ['On-call team', 'Management'],
                    'method': 'PagerDuty + Phone',
                    'message': 'Initial alert',
                    'timeframe': '<5 minutes'
                },
                '15_minutes': {
                    'recipients': ['Extended team', 'Directors'],
                    'method': 'Slack + Email',
                    'message': 'Situation update',
                    'timeframe': '<15 minutes'
                },
                '30_minutes': {
                    'recipients': ['All staff', 'Executives'],
                    'method': 'All-hands call',
                    'message': 'Detailed briefing',
                    'timeframe': '<30 minutes'
                },
                'hourly': {
                     'recipients': ['Board', 'Investors'],
                     'method': 'Email + Call',
                     'message': 'Executive summary',
                     'timeframe': 'Every hour'
                 }
             },

             'external_notifications': {
                 'customers': {
                     'trigger': 'Service impact >5 minutes',
                     'method': 'Status page + Email',
                     'message_template': '''
                         Subject: MWRASP Service Disruption
Notification

                         We are currently experiencing [ISSUE].
                         Impact: [DESCRIPTION]
                         Started: [TIME]
                         Expected Resolution: [ETA]

                         Updates: [STATUS_PAGE_URL]
                     ''',
                     'frequency': 'Every 30 minutes'
                 },

                 'regulatory': {
                     'trigger': 'Data breach or extended outage',
                     'agencies': ['SEC', 'GDPR authorities', 'HIPAA'],
                     'timeframe': 'Within 72 hours',
                     'method': 'Formal notification'
                 },

                 'media': {
                     'trigger': 'Public visibility',
                     'response': 'Prepared statement',
                     'spokesperson': 'CEO or designated',
                     'channels': ['Press release', 'Social media']
                 }
             }
         }

     def communication_templates(self) -> Dict:
         """
         Pre-written communication templates
         """
         return {
             'initial_notification': {
                 'internal': '''
                 URGENT: System Incident Detected

                 Severity: [LEVEL]
                 Systems Affected: [SYSTEMS]
```

```
                Initial Impact: [IMPACT]
                Response Team: Activated

                Join crisis bridge: [BRIDGE_URL]
                ''',
                'external': '''
                Service Disruption Notice

                We are investigating an issue affecting [SERVICE].
                Some users may experience [IMPACT].
                Our team is working to resolve this quickly.

                Updates: [STATUS_URL]
                '''
            },

            'progress_update': {
                'internal': '''
                Incident Update - [TIME]

                Current Status: [STATUS]
                Progress: [PROGRESS]
                Next Steps: [ACTIONS]
                ETA: [ESTIMATE]
                ''',
                'external': '''
                Service Update - [TIME]

                We continue to work on resolving the issue.
                Current status: [STATUS]
                Expected resolution: [ETA]

                We apologize for any inconvenience.
                '''
            },

            'resolution_notice': {
                'internal': '''
                Incident Resolved

                Resolution Time: [TIME]
                Root Cause: [CAUSE]
                Services Restored: [SERVICES]
                Follow-up Actions: [ACTIONS]

                Post-mortem scheduled: [DATE]
                ''',
                'external': '''
                Service Restored

                The issue has been resolved as of [TIME].
                All services are now operational.
                We apologize for the disruption and thank you
                for your patience.
                '''
            }
        }
```

SECTION 6: POST-INCIDENT PROCEDURES

6.1 Post-Incident Review

```
class PostIncidentReview:
    """
    Post-incident analysis and improvement
    """

    def __init__(self):
        self.review_timeline = '48 hours post-incident'

    def post_mortem_process(self) -> Dict:
        """
        Comprehensive post-mortem process
        """
        return {
            'immediate_actions': {
                'timeline': 'Within 24 hours',
                'tasks': [
                    'Preserve all logs and data',
                    'Document timeline of events',
                    'Capture team observations',
                    'Identify immediate fixes'
                ]
            },

            'post_mortem_meeting': {
                'timeline': 'Within 48 hours',
                'duration': '2 hours',
                'participants': [
                    'Incident response team',
                    'Technical leads',
                    'Management',
                    'Customer success'
                ],
                'agenda': [
                    'Timeline review',
                    'Root cause analysis',
                    'Response evaluation',
                    'Improvement identification',
                    'Action items assignment'
                ]
            },

            'root_cause_analysis': {
                'methodology': 'Five Whys + Fishbone',
                'categories': [
                    'Technical factors',
                    'Process factors',
                    'Human factors',
                    'External factors'
                ],
                'deliverables': [
                    'Root cause identification',
                    'Contributing factors',
                    'Prevention recommendations',
                    'Risk assessment update'
                ]
            },

            'improvement_plan': {
                'categories': {
                    'immediate': {
                        'timeline': '<1 week',
                        'examples': [
                            'Configuration changes',
                            'Monitoring additions',
                            'Documentation updates'
                        ]
                    },
                    'short_term': {
                        'timeline': '1-4 weeks',
                        'examples': [
                            'Process improvements',
                            'Training programs',
                            'Tool enhancements'
                        ]
                    },
                    'long_term': {
                        'timeline': '1-6 months',
                        'examples': [
                            'Architecture changes',
                            'Capacity upgrades',
                            'New technologies'
                        ]
                    }
                }
            },

            'documentation': {
                'incident_report': {
```

```
                'sections': [
                    'Executive summary',
                    'Timeline of events',
                    'Impact assessment',
                    'Root cause analysis',
                    'Response evaluation',
                    'Lessons learned',
                    'Action items'
                ],
                'distribution': [
                    'Executive team',
                    'Board (if major)',
                    'Key customers (sanitized)',
                    'Internal teams'
                ]
            },

            'knowledge_base_update': [
                'Update runbooks',
                'Revise procedures',
                'Add new scenarios',
                'Update training materials'
            ]
        }
    }
```

SECTION 7: COMPLIANCE AND AUDIT

7.1 Regulatory Requirements

```
class RegulatoryCompliance:
    """
    DR compliance with regulations
    """

    def __init__(self):
        self.regulations = ['SOC2', 'ISO22301', 'GDPR', 'HIPAA']

    def compliance_requirements(self) -> Dict:
        """
        Regulatory requirements for DR
        """
        return {
            'SOC2_requirements': {
                'availability': {
                    'requirement': '99.9% uptime',
                    'evidence': 'Uptime reports',
                    'testing': 'Annual DR test'
                },
                'processing_integrity': {
                    'requirement': 'Complete and accurate',
                    'evidence': 'Data validation logs',
                    'testing': 'Integrity checks'
                },
                'confidentiality': {
                    'requirement': 'Data protection',
                    'evidence': 'Encryption verification',
                    'testing': 'Security audits'
                }
            },

            'ISO22301_requirements': {
                'business_continuity': {
                    'bcp_documentation': 'Complete BCP plan',
                    'risk_assessment': 'Annual update',
                    'testing': 'Regular exercises',
                    'management_review': 'Quarterly'
                },
                'recovery_objectives': {
                    'rto_documentation': 'Defined and tested',
                    'rpo_documentation': 'Defined and tested',
                    'mtpd': 'Maximum tolerable period'
                }
            },

            'GDPR_requirements': {
                'data_breach_notification': {
                    'timeframe': '72 hours',
                    'authorities': 'Supervisory authority',
                    'individuals': 'If high risk',
                    'documentation': 'Breach register'
                },
                'data_recovery': {
                    'backup_requirements': 'Secure and encrypted',
                    'restoration': 'Timely manner',
                    'integrity': 'Maintained'
                }
            },

            'audit_evidence': {
                'test_results': 'All DR test documentation',
                'incident_reports': 'Post-mortem reports',
                'training_records': 'Staff preparedness',
                'procedure_updates': 'Version controlled',
                'executive_reviews': 'Board presentations'
            }
        }
```

SECTION 8: RECOVERY RUNBOOKS

8.1 Service-Specific Runbooks

```
 #!/bin/bash
# Quantum Canary Recovery Runbook

# RUNBOOK: Quantum Canary Service Recovery
# SEVERITY: CRITICAL
# ESTIMATED TIME: 30 minutes
# DEPENDENCIES: Core infrastructure must be operational

echo "========================================="
echo "Quantum Canary Service Recovery Runbook"
echo "Started at: $(date)"
echo "========================================="

# Step 1: Verify Infrastructure
echo "[Step 1] Verifying infrastructure..."
check_infrastructure() {
    # Check network connectivity
    if ! ping -c 1 quantum-controller.internal >/dev/null 2>&1; then
        echo "ERROR: Cannot reach quantum controller"
        exit 1
    fi

    # Check storage availability
    if ! df -h | grep -q "/var/lib/quantum"; then
        echo "ERROR: Quantum storage not mounted"
        exit 1
    fi

    echo "  Infrastructure verified"
}

# Step 2: Restore Quantum Canary Configuration
echo "[Step 2] Restoring configuration..."
restore_configuration() {
    # Restore from backup
    latest_backup=$(ls -t /backup/quantum-canary/*.tar.gz | head -1)

    if [ -z "$latest_backup" ]; then
        echo "ERROR: No backup found"
        exit 1
    fi

    tar -xzf "$latest_backup" -C /etc/quantum-canary/
    echo "  Configuration restored from $latest_backup"
}
```

```
# Step 3: Initialize Quantum Entanglement
echo "[Step 3] Initializing quantum entanglement..."
initialize_entanglement() {
    python3 <<EOF
import sys
sys.path.append('/opt/mwrasp/lib')
from quantum_canary import QuantumCanarySystem

qcs = QuantumCanarySystem()
qcs.initialize_entanglement()
qcs.deploy_canaries(count=100)
print("  Quantum entanglement established")
EOF
}

# Step 4: Start Canary Services
echo "[Step 4] Starting canary services..."
start_services() {
    systemctl start quantum-canary-controller
    systemctl start quantum-canary-monitor
    systemctl start quantum-canary-alert

    # Wait for services to be ready
    sleep 10

    # Verify services
    for service in controller monitor alert; do
        if ! systemctl is-active quantum-canary-$service >/dev/null;
then
            echo "ERROR: Service quantum-canary-$service failed to
start"
            exit 1
        fi
    done

    echo "  All canary services started"
}

# Step 5: Validate Recovery
echo "[Step 5] Validating recovery..."
validate_recovery() {
    # Test canary detection
    curl -s http://localhost:8443/api/v1/canary/test

    # Check canary status
    canary_status=$(curl -s http://localhost:8443/api/v1/canary/status
| jq -r '.status')

    if [ "$canary_status" != "operational" ]; then
        echo "ERROR: Canaries not operational"
        exit 1
    fi

    echo "  Recovery validated"
}

# Step 6: Update Monitoring
echo "[Step 6] Updating monitoring..."
update_monitoring() {
    # Send recovery notification
    curl -X POST https://monitoring.internal/api/v1/events \
        -H "Content-Type: application/json" \
        -d '{
            "event": "quantum_canary_recovered",
            "timestamp": "'$(date -Iseconds)'",
            "details": "Service recovered successfully"
        }'

    echo "  Monitoring updated"
}

# Main execution
main() {
    check_infrastructure
    restore_configuration
    initialize_entanglement
    start_services
    validate_recovery
    update_monitoring

    echo "=========================================="
    echo "Recovery completed successfully"
    echo "Completed at: $(date)"
    echo "=========================================="
}

# Run with error handling
set -e
trap 'echo "ERROR: Recovery failed at line $LINENO"' ERR
main
```

APPENDIX A: CONTACT LISTS

Emergency Contacts

Role	Name	Primary Phone	Backup Phone	Email
Incident Commander	John Smith	+1-555-0100	+1-555-0101	john.smith@mwrasp.com
Technical Lead	Sarah Chen	+1-555-0102	+1-555-0103	sarah.chen@mwrasp.com
Communications Lead	Mike Johnson	+1-555-0104	+1-555-0105	mike.johnson@mwrasp.com
Executive Sponsor	Lisa Wang	+1-555-0106	+1-555-0107	lisa.wang@mwrasp.com

Vendor Contacts

Vendor	Service	Support Number	Account #
AWS	Cloud Infrastructure	1-800-xxx-xxxx	12345
Datadog	Monitoring	1-866-xxx-xxxx	67890
PagerDuty	Alerting	1-844-xxx-xxxx	11111

APPENDIX B: RECOVERY CHECKLISTS

Quick Recovery Checklist

- [ ] Incident confirmed
- [ ] Response team activated
- [ ] Communication bridge opened
- [ ] Initial assessment complete
- [ ] Recovery strategy selected
- [ ] Failover initiated (if needed)
- [ ] Services being restored

- [ ] Stakeholders notified
- [ ] Monitoring active
- [ ] Validation in progress
- [ ] Normal operations resumed
- [ ] Post-mortem scheduled

---

*End of Disaster Recovery Procedures Classification: Confidential * 2025 MWRASP Quantum Defense System*
