瀏覽器操作員功能驗收清單與工作手冊 v1.0\n\n## 執行摘要\n\n本文件提供了瀏覽器操作員一站式集成系統的完整功能驗收清單與具體操作示例。清單涵蓋核心功能、安全機制、性能指標、合規要求與運維能力的詳細驗證步驟。所有項目均與之前定義的規則集、架構與實現直接對應。\n\n---\n\n## 第一部分：部署前驗收\n\n### 基礎設施驗收清單\n\n| 項目 | 驗收標準 | 檢查方法 | 狀態 | 簽名 | 日期 |\n|---|---|---|---|---|---|\n| Kubernetes 集群 | 3+ Master, 3+ Worker 節點 | `kubectl get nodes` | ☐ | __ | __ |\n| 網路連通性 | 所有節點 ping 通，延遲 < 5ms | `ping`, `traceroute` | ☐ | __ | __ |\n| 儲存配置 | Redis Cluster, Elasticsearch, PostgreSQL | `kubectl get pvc` | ☐ | __ | __ |\n| 外部服務 | CloudHSM, CloudWatch, Vault 可達 | `telnet`, `aws service check` | ☐ | __ | __ |\n| DNS 解析 | 所有服務名解析正確 | `nslookup`, `dig` | ☐ | __ | __ |\n| TLS 證書 | 有效期 > 365 天 | `openssl x509 -text` | ☐ | __ | __ |\n| 防火牆規則 | 必要端口開放 | 檢查安全組規則 | ☐ | __ | __ |\n| 容量規劃 | 磁碟 > 1TB, RAM > 32GB | `df -h`, `free -h` | ☐ | __ | __ |\n| 備份系統 | S3, 每日自動備份 | 驗證備份腳本 | ☐ | __ | __ |\n| 監控系統 |

Prometheus, Grafana 可訪問 | 訪問儀表板 | □ | __ | __ |\n\n### 組件健康檢查\n\n bash\n#!/bin/bash\n# health_check.sh - 全面的健康檢查腳本\n\necho \"=== Kubernetes 集群檢查 ===\"\nkubectl cluster-info\nkubectl get nodes -o wide\n\necho \"\\n=== Pod 狀態檢查 ===\"\nkubectl get pods -n browser-operator -o wide\n\necho \"\\n=== Redis 檢查 ===\"\nredis-cli -h redis-master ping\nredis-cli -h redis-master info replication\nredis-cli -h redis-master --latency\n\necho \"\\n=== Elasticsearch 檢查 ===\"\ncurl -s http://elasticsearch:9200/_cluster/health | jq .\n\necho \"\\n=== PostgreSQL 檢查 ===\"\npsql -h postgres-master -U browser_operator -c \"SELECT version( );\"\n\necho \"\\n=== 密鑰管理檢查 ===\"\naws cloudhsm describe-clusters --query 'Clusters[0].HsmType'\n\necho \"\\n=== 全部檢查完成 ===\"\n \n\n---\n\n## 第二部分：功能驗收清單\n\n### 認證與授權功能\n\n測試用例 1：MFA 認證流程\n\n bash\n# 測試場景：使用者使用 MFA 建立會話\n\n# 1. 準備測試資料\nUSER_ID=\"test_user_001\"\nIP_ADDRESS=\"192.168.1.100\"\nMFA_TOKEN=\"123456\"\n\n# 2. 呼叫認證 API\ncurl -X POST http://api-

server:8080/api/v1/auth/authenticate \\\n -H \"Content-Type: application/json\" \\\n -d \"{\n \\\"user_id\\\": \\\"$USER_ID\\\",\n \\\"ip_address\\\": \\\"$IP_ADDRESS\\\",\n \\\"mfa_token\\\": \\\"$MFA_TOKEN\\\"\n }\"\n\n# 預期結果：\n# {\"session_id\": \"sess_xxx\", \"bearer_token\": \"eyJ...\", \"status\": \"success\"}\n\n# 3. 驗收標準\n# ✓ 返回狀態 200\n# ✓ Response 包含 session_id 和 bearer_token\n# ✓ Token 有效期 30 分鐘\n# ✓ 審計日誌記錄該事件\n \n\n| 驗收項 | 標準 | 實際結果 | 狀態 |\n|---|---|---|---|\n| API 響應時間 | < 100 ms | __ | ☐ |\n| MFA 驗證成功率 | 100% | __ | ☐ |\n| 會話建立成功 | 100% | __ | ☐ |\n| 審計日誌完整性 | 100% 記錄 | __ | ☐ |\n| Token 簽名有效 | 100% 通過驗證 | __ | ☐ |\n\n測試用例 2：RBAC 授權檢查\n\n bash\n# 測試場景：驗證不同角色的權限限制\n\n# 1. 使用 VIEWER 角色嘗試執行操作\nSESSION_ID=\"sess_viewer_001\"\nTOKEN=\"bearer_token_xxx\"\n\ncurl -X POST http://api-server:8080/api/v1/operator/execute \\\n -H \"Authorization: Bearer $TOKEN\" \\\n -H \"Content-Type: application/json\" \\\n -d \"{\n \\\"session_id\\\": \\\"$SESSION_ID\\\",\n \\\"operation\\\": \\\"execute_script\\\",\n

\\\"resource\\\": \\\"browser_session\\\"\n }\"\n\n# 預期結果：\n# {\"error\": \"Operation not permitted for role VIEWER\", \"status\": 403}\n\n# 2. 使用 OPERATOR 角色執行相同操作\nSESSION_ID=\"sess_operator_001\"\n\ncurl -X POST http://api-server:8080/api/v1/operator/execute \\\n -H \"Authorization: Bearer $TOKEN\" \\\n -H \"Content-Type: application/json\" \\\n -d \"{\n \\\"session_id\\\": \\\"$SESSION_ID\\\",\n \\\"operation\\\": \\\"execute_script\\\",\n \\\"resource\\\": \\\"browser_session\\\"\n }\"\n\n# 預期結果：\n# {\"status\": \"success\", \"operation_id\": \"op_xxx\"}\n \n\n| 驗收項 | 標準 | 實際結果 | 狀態 |\n|---|---|---|---|\n| VIEWER 拒絕執行 | 返回 403 | __ | ☐ |\n| OPERATOR 允許執行 | 返回 200 | __ | ☐ |\n| ADMIN 全部允許 | 返回 200 | __ | ☐ |\n| 拒絕記錄審計 | 100% 記錄 | __ | ☐ |\n| 權限層級正確 | 順序正確 | __ | ☐ |\n\n### 操作完整性與驗證\n\n測試用例 3：原子性操作\n\n bash\n# 測試場景：驗證操作的原子性（全部成功或全部失敗）\n\n# 1. 準備一個將會失敗的操作\nOPERATION_ID=\"op_atomic_001\"\n\ncurl -X POST http://api-

server:8080/api/v1/operator/execute \\\n -H \"Authorization: Bearer $TOKEN\" \\\n -H \"Content-Type: application/json\" \\\n -d \"{\n \\\"session_id\\\": \\\"$SESSION_ID\\\",\n \\\"operation\\\": \\\"navigate\\\",\n \\\"resource\\\": \\\"https://nonexistent-domain-12345.com\\\",\n \\\"timeout_ms\\\": 5000\n }\"\n\n# 預期結果：操作失敗\n# {\"status\": \"failed\", \"error\": \"Connection timeout\"}\n\n# 2. 驗證系統狀態已回滾\nkubectl exec -it <redis-pod> -- redis-cli \\\n GET \"operation:state:$OPERATION_ID\"\n\n# 預期結果：應返回 nil（表示已清理）或 ROLLED_BACK 狀態\n \n\n| 驗收項 | 標準 | 實際結果 | 狀態 |\n|---|---|---|---|\n| 成功操作完成 | 100% 成功 | __ | ☐ |\n| 失敗操作回滾 | 100% 回滾 | __ | ☐ |\n| 狀態一致性 | 無孤立狀態 | __ | ☐ |\n| 結果驗證執行 | 100% 驗證 | __ | ☐ |\n| 超時自動回滾 | >1 秒回滾 | __ | ☐ |\n\n### 安全與隔離功能\n\n測試用例 4：異常檢測與自動隔離\n\n bash\n# 測試場景：系統檢測異常行為並自動隔離\n\n# 1. 模擬異常操作頻率（強制循環執行）\nfor i in {1..150}; do\n curl -X POST http://api-server:8080/api/v1/operator/execute \\\n -H \"Authorization: Bearer $TOKEN\" \\\n -H

\"Content-Type: application/json\" \\\n -d \"{\n \\\"session_id\\\": \\\"$SESSION_ID\\\",\n \\\"operation\\\": \\\"get_status\\\",\n \\\"resource\\\": \\\"test\\\"\n }\"\n sleep 0.1\ndone\n\n# 預期結果：\n# 前 100 次成功 ，第 101-150 次被速率限制\n# 響應 429: {\"error\": \"Rate limit exceeded\"}\n\n# 2. 驗證使用者狀態\nkubectl exec -it <redis-pod> -- redis-cli \\\n GET \"user:supervision:$USER_ID\"\n\n# 預期結果：應返回監督模式配置\n \n\n| 驗收項 | 標準 | 實際結果 | 狀態 |\n|---|---|---|---|\n| 異常檢測延遲 | < 1 秒 | __ | ☐ |\n| 自動限流啟動 | 立即啟動 | __ | ☐ |\n| 降速應用 | 50% 容量 | __ | ☐ |\n| 隔離紀錄 | 100% 記錄 | __ | ☐ |\n| 升報通知 | 立即發送 | __ | ☐ |\n\n測試用例 5：職責分離 (SoD) 強制\n\n bash\n# 測試場景：防止同人執行衝突操作\n\n# 1. 操作者 A 發起一個部署請求\nUSER_A=\"operator_a\"\nDEPLOY_REQUEST_ID=\"req_deploy_001\"\n\n# 2. 嘗試同一用戶立即批准（應被拒絕）\ncurl -X POST http://api-server:8080/api/v1/operator/approve \\\n -H \"Authorization: Bearer $TOKEN_A\" \\\n -H \"Content-Type: application/json\" \\\n -d \"{\n \\\"request_id\\\": \\\"$DEPLOY_REQUEST_ID\\\",\n

\\\"approver\\\": \\\"$USER_A\\\"\n }\"\n\n# 預期結果：\n# {\"error\": \"SoD violation: User cannot request and approve same operation\", \"status\": 403}\n\n# 3. 操作者 B 批准（應成功）\ncurl -X POST http://api-server:8080/api/v1/operator/approve \\\n -H \"Authorization: Bearer $TOKEN_B\" \\\n -H \"Content-Type: application/json\" \\\n -d \"{\n \\\"request_id\\\": \\\"$DEPLOY_REQUEST_ID\\\",\n \\\"approver\\\": \\\"$USER_B\\\"\n }\"\n\n# 預期結果：\n# {\"status\": \"success\", \"approval_id\": \"appr_xxx\"}\n \n\n| 驗收項 | 標準 | 實際結果 | 狀態 |\n|---|---|---|---|\n| SoD 檢測 | 100% 檢測 | __ | ☐ |\n| 同人衝突拒絕 | 100% 拒絕 | __ | ☐ |\n| 不同人批准 | 100% 允許 | __ | ☐ |\n| 衝突記錄 | 100% 記錄 | __ | ☐ |\n| 升報管理 | 立即升報 | __ | ☐ |\n\n### 審計與日誌功能\n\n測試用例 6：不可篡改的審計鏈\n\n bash\n# 測試場景：驗證審計日誌的密碼學完整性\n\n# 1. 執行操作並記錄審計 ID\nOPERATION_ID=\"op_audit_001\"\n# ... 執行某個操作\n\n# 2. 檢索審計日誌\nAUDIT_ID=$(curl -s http://api-server:8080/api/v1/audit/logs \\\n -H \"Authorization: Bearer $TOKEN\" \\\n -H \"Content-Type: application/json\" \\\n -d \"

{\\\"operation_id\\\": \\\"$OPERATION_ID\\\"}\" | jq -r '.logs[0].log_id')\n\n# 3. 獲取完整日誌記錄\ncurl -s http://api-server:8080/api/v1/audit/logs/$AUDIT_ID \\\n -H \"Authorization: Bearer $TOKEN\" > /tmp/audit_log.json\n\n# 4. 驗證簽名\necho \"驗證簽名...\"\nkubectl exec -it <api-pod> -- python -c \"\nimport json\nwith open('/tmp/audit_log.json') as f:\n log = json.load(f)\nfrom crypto_service import CryptoService\ncs = CryptoService()\nresult = cs.verify_signature(\n json.dumps(log, sort_keys=True).encode(),\n log['signature']\n)\nprint(f'簽名驗證: {\\\"✓ 有效\\\" if result else \\\"✗ 無效\\\"}')\n\"\n\n# 5. 嘗試篡改日誌（應被檢測）\njq '.details.status = \\\"MODIFIED\\\"' /tmp/audit_log.json > /tmp/audit_log_modified.json\n\necho \"驗證已修改的日誌...\"\nkubectl exec -it <api-pod> -- python -c \"\nimport json\nwith open('/tmp/audit_log_modified.json') as f:\n log = json.load(f)\nfrom crypto_service import CryptoService\ncs = CryptoService()\nresult = cs.verify_signature(\n json.dumps(log, sort_keys=True).encode(),\n log['signature']\n)\nprint(f'簽名驗證: {\\\"✓ 有效\\\"

if result else \\\"✗ 無效 (已篡改)\\\"}')\n\"\n \n\n| 驗收項 | 標準 | 實際結果 | 狀態 |\n|-------|------|--------|------|\n| 日誌雜湊計算 | 正確 SHA3-512 | __ | ☐ |\n| 簽名生成 | RSA-4096 有效 | __ | ☐ |\n| 簽名驗證 | 100% 通過 | __ | ☐ |\n| 篡改檢測 | 100% 檢測 | __ | ☐ |\n| 鏈式完整性 | 序列號無缺失 | __ | ☐ |\n\n測試用例 7：日誌保留與銷毀政策\n\n bash\n# 測試場景：驗證日誌自動銷毀機制\n\n# 1. 建立 LOW 級別日誌（保留 90 天）\n# ... 記錄 LOW 級別操作\n\n# 2. 檢查保留政策\ncurl -s http://api-server:8080/api/v1/audit/retention-policy \\\n -H \"Authorization: Bearer $TOKEN\" |jq .\n\n# 預期結果包含：\n# {\n# \"LOW\": {\"retention_days\": 90, \"method\": \"3-pass_overwrite\"},\n# \"MEDIUM\": {\"retention_days\": 365, \"method\": \"5-pass_overwrite\"},\n# ...\n# }\n\n# 3. 驗證銷毀任務排程\nkubectl logs -f deployment/audit-service | grep \"destruction_scheduled\"\n \n\n| 驗收項 | 標準 | 實際結果 | 狀態 |\n|-------|------|--------|------|\n| LOW 級日誌 | 90 天後銷毀 | __ | ☐ |\n| MEDIUM 級日誌 | 1 年後銷毀 | __ | ☐ |\n| CRITICAL 級日誌 | 10 年保留 | __ | ☐ |\n| 銷毀驗證 | 3-7 次覆蓋 | __ | ☐ |\n| 銷毀記錄 | 100% 記錄 | __ | ☐ |\n\n---\n\n## 第三部分：性能驗收\n\n### 性能基準測試\n\n測試用例 8：API 響

應延遲\n\n```bash\n#!/bin/bash\n# performance_test.sh - 性能基準測試\n\necho \"開始性能測試...\"\n\n# 1. 單個 API 呼叫延遲\necho \"測試單個 API 響應時間...\"\nfor i in {1..1000}; do\n time curl -s http://api-server:8080/api/v1/health\ndone | grep real | awk '{print $2}' | tee /tmp/latencies.txt\n\n# 2. 計算統計量\necho \"\\n統計信息:\"\necho \"平均延遲 (ms): $(awk '{sum+=$1; n++} END {print sum/n*1000}' /tmp/latencies.txt)\"\necho \"P95 延遲: $(sort -n /tmp/latencies.txt | awk '{a[NR]=$1} END {print a[int(NR*0.95)]*1000}')\"\necho \"P99 延遲: $(sort -n /tmp/latencies.txt | awk '{a[NR]=$1} END {print a[int(NR*0.99)]*1000}')\"\necho \"最大延遲: $(sort -n /tmp/latencies.txt | tail -1 | awk '{print $1*1000}')\"\n\n# 3. 並發測試\necho \"\\n\\n測試並發操作...\"\nab -n 10000 -c 100 http://api-server:8080/api/v1/health\n```\n\n| 指標 | 目標 | 測試結果 | 狀態 |\n|-----|------|--------|------|\n| 平均延遲 | < 150 ms | __ | ☐ |\n| P95 延遲 | < 250 ms | __ | ☐ |\n| P99 延遲 | < 500 ms | __ | ☐ |\n| 最大延遲 | < 1000 ms | __ | ☐ |\n| 吞吐量 | > 1000 req/s | __ | ☐ |\n| 錯誤率 | < 0.1% | __ | ☐ |\n\n### 測試用例 9：資源使用效率\n\n```bash\n# 測試場景：監控系統資源使用\n\necho \"監控資源使用 1 小

時...\"\n\nfor i in {1..60}; do\n echo \"分鐘 $i:\"\n \n # 1. CPU 使用\n kubectl top nodes | tail -n +2\n \n # 2. 記憶體使用\n kubectl top pods -n browser-operator\n \n # 3. Redis 記憶體\n redis-cli -h redis-master info memory | grep used_memory_human\n \n # 4. Elasticsearch 磁碟\n curl -s http://elasticsearch:9200/_cat/allocation?v\n \n sleep 60\ndone | tee /tmp/resource_usage.log\n \n\n| 資源 | 目標 | 實際 | 狀態 |\n|-----|------|------|------|\n| CPU 使用率 | < 60% | __ | ☐ |\n| 記憶體使用率 | < 70% | __ | ☐ |\n| 磁碟使用率 | < 80% | __ | ☐ |\n| 網路頻寬 | < 50% | __ | ☐ |\n| I/O 延遲 | < 50 ms | __ | ☐ |\n\n---\n\n## 第四部分：合規驗收\n\n### 合規框架驗證\n\n測試用例 10：SOC 2 控制驗證\n\n bash\necho \"檢查 SOC 2 合規控制...\"\n\n# 1. 存取控制\necho \"\\n1. 存取控制驗證:\"\necho \" ✓ MFA 強制: $(curl -s http://api-server:8080/api/v1/compliance/mfa-status |jq '.enforced')\"\necho \" ✓ RBAC 實施: $(curl -s http://api-server:8080/api/v1/compliance/rbac-status |jq '.enforced')\"\n\n# 2. 審計日誌\necho \"\\n2. 審計日誌驗證:\"\necho \" ✓ 日誌完整性: $(curl -s http://api-server:8080/api/v1/compliance/audit-integrity |jq '.status')\"\necho \" ✓ 日誌保留: $(curl -

s http://api-server:8080/api/v1/compliance/retention-policy | jq '.critical_retention')\"\n\n# 3. 加密\necho \"\\n3. 加密驗證:\"\necho \" ✓ TLS 版本: $(openssl s_client -connect api-server:443 </dev/null 2>&1 | grep 'Protocol')\"\necho \" ✓ 金鑰強度: $(curl -s http://api-server:8080/api/v1/compliance/crypto-status | jq '.key_strength')\"\n\n# 4. 可用性\necho \"\\n4. 可用性驗證:\"\necho \" ✓ SLA 合規: $(curl -s http://api-server:8080/api/v1/compliance/sla-status | jq '.compliant')\"\necho \" ✓ 備份狀態: $(curl -s http://api-server:8080/api/v1/compliance/backup-status | jq '.last_backup')\"\n \n\n| 控制 | 要求 | 驗證結果 | 狀態 |\n|-----|------|--------|------|\n| CC6.1 - 邏輯和物理存取 | 已實施 | __ | ☐ |\n| CC7.2 - 系統監控 | 已實施 | __ | ☐ |\n| CC7.3 - 評估和報告 | 已實施 | __ | ☐ |\n| A1.1 - 可用性目標 | 99.9% | __ | ☐ |\n| A1.2 - 可用性達成 | > 99.9% | __ | ☐ |\n\n 測試用例 11：GDPR 合規驗證\n\n bash\necho \"檢查 GDPR 合規要求...\"\n\n# 1. 資料最小化\necho \"\\n1. 資料最小化:\"\necho \" ✓ 收集必要性: $(curl -s http://api-server:8080/api/v1/compliance/data-minimization | jq '.status')\"\n\n# 2. 用戶權利\necho \"\\n2. 用戶權利:\"\necho \" ✓ 存取請求:

```
$(curl -s http://api-server:8080/api/v1/compliance/data-access-capability | jq '.enabled' )\"\necho \" ✓ 刪除請求: $(curl -s http://api-server:8080/api/v1/compliance/data-deletion-capability | jq '.enabled' )\"\necho \" ✓ 可移植性: $(curl -s http://api-server:8080/api/v1/compliance/data-portability | jq '.enabled' )\"\n\n# 3. 隱私保護\necho \"\\n3. 隱私保護:\"\necho \" ✓ 加密傳輸: $(curl -s http://api-server:8080/api/v1/compliance/encryption-in-transit | jq '.enabled' )\"\necho \" ✓ 加密存儲: $(curl -s http://api-server:8080/api/v1/compliance/encryption-at-rest | jq '.enabled' )\"\n \n\n| 要求 | 實施狀態 | 驗證結果 | 狀態 |\n|-----|--------|--------|------|\n| 第 4 條 - 定義和原則 | 已實施 | __ | ☐ |\n| 第 5 條 - 資料最小化 | 已實施 | __ | ☐ |\n| 第 6 條 - 合法基礎 | 已實施 | __ | ☐ |\n| 第 15-21 條 - 用戶權利 | 已實施 | __ | ☐ |\n\n---\n\n## 第五部分：簽收與移交\n\n### 最終驗收簽名頁\n\n系統部署驗收表\n\n \n系統名稱: 瀏覽器操作員一站式集成系統 v1.0\n部署日期: _____\n部署環境: ☐ 開發 ☐ 測試 ☐ 預發布 ☐ 生產\n\n核心功能驗收:\n ☐ 認證與授權功能：已通過 (簽名：____ 日期：___)\n ☐ 操作完整性與驗證：已通過 (簽
```

名：____ 日期：___)\n ☐ 安全隔離功能：已通過 (簽名：____ 日期：___)\n ☐ 審計與日誌功能：已通過 (簽名：____ 日期：___)\n\n性能驗收:\n ☐ API 響應延遲：已通過 (簽名：____ 日期：___)\n ☐ 資源效率：已通過 (簽名：____ 日期：___)\n ☐ 可擴展性：已通過 (簽名：____ 日期：___)\n\n合規驗收:\n ☐ SOC 2 控制：已通過 (簽名：____ 日期：___)\n ☐ GDPR 要求：已通過 (簽名：____ 日期：___)\n ☐ 內部政策：已通過 (簽名：____ 日期：___)\n\n運維準備:\n ☐ 文檔完整：已完成 (簽名：____ 日期：___)\n ☐ 團隊培訓：已完成 (簽名：____ 日期：___)\n ☐ 備份測試：已通過 (簽名：____ 日期：___)\n\n整體驗收:\n ☐ 系統已就緒，可投入生產 (簽名：____ 日期：___)\n \n質量保證負責人: _____ 簽名：_____ 日期：___\n項目經理: _____ 簽名：_____ 日期：___\n系統架構師: _____ 簽名：_____ 日期：___\n業務負責人: _____ 簽名：_____ 日期：___\n \n\n---\n\n版本: 1.0 \n維護者: IndestructibleAutoOps QA & Operations Team \n最後更新: 2026 年 2 月 5 日\n