

Drop Srbija v2 — Runbooks

Drop Srbija v2 — Runbooks

Purpose: Operational procedures for common incidents and tasks.

Runbook 1: NBS IPS Outage

Source: Existing runbook (kept from v1)

File: docs/04-runbook-nbs-ips-outage.md

Symptoms

- `POST /v1/ips/initiate` returns 503 Service Unavailable
- Sentry alert: "NBSIPSDown" (NBS API unreachable > 5 min)
- Grafana: NBS IPS p95 latency > 10s
- User reports: "Payment failed, try again later"

Triage

1. **Check NBS IPS status page:** <https://ips.nbs.rs/status> (if exists)
2. **Check NBS API health:**

```
curl -I https://ips.nbs.rs/api/v1/health
```

3. **Check audit_log for recent NBS IPS calls:**

```
SELECT * FROM nbs_ips_logs
WHERE timestamp > NOW() - INTERVAL '1 hour'
ORDER BY timestamp DESC LIMIT 10;
```

Resolution

If NBS IPS is down (confirmed):

1. Enable maintenance mode:

```
curl -X POST https://drop.rs/v1/admin/maintenance \
  -H "Authorization: Bearer $ADMIN_TOKEN" \
  -d '{"enabled": true, "message": "NBS IPS privremeno nedostupan. Molimo pokušajte za 30 minuta."}'
```

2. Notify users (push notification + in-app banner):

- Message: "NBS IPS trenutno nije dostupan. Uplate će biti omogućene čim sistem bude aktivan."

3. Queue pending transactions (do NOT reject):

- Backend automatically queues transactions when NBS IPS returns 5xx
- Queue retention: 4 hours (after that, user must retry)

4. Monitor NBS IPS recovery:

- Check status page every 15 min
- When NBS IPS returns 200 OK, disable maintenance mode

5. Process queued transactions:

```
curl -X POST https://drop.rs/v1/admin/queue/process \
  -H "Authorization: Bearer $ADMIN_TOKEN"
```

RTO: 0 minutes (graceful degradation, no data loss)

Runbook 2: Backup Recovery

Source: `docs/runbooks/backup-recovery.md`

Scenario: Database Corruption

Symptoms:

- Backend crashes with "Database connection failed"
- Sentry: "PostgreSQL constraint violation"
- Data integrity check fails

Steps:

1. Stop backend container apps:

```
az containerapp stop --name dropsrbija-backend --resource-group dropsrbija-prod
```

2. Restore PostgreSQL from latest backup:

```
az postgres flexible-server restore \  
  --resource-group dropsrbija-prod \  
  --name dropsrbija-db-restored \  
  --source-server dropsrbija-db \  
  --restore-time "2026-04-17T03:00:00Z" # Latest backup
```

3. Verify data integrity:

```
psql -h dropsrbija-db-restored.postgres.database.azure.com \  
  -U dropsrbija_admin -d dropsrbija_prod \  
  -c "SELECT COUNT(*) FROM users; SELECT COUNT(*) FROM transactions;"
```

4. Swap DNS to restored database:

- Update `DATABASE_URL` in Key Vault
- Restart backend container apps

5. Monitor error logs:

- Sentry: Check for new errors
- Grafana: Check error rate dashboard

RTO: 4 hours

Runbook 3: Security Incident Response

Trigger: ZPNFTM 4-hour notification + 72-hour NBS report + 72-hour Poverenik report

ZPNFTM (Law on Payment Services) Requirements

Article 108: Incident Reporting

- **Initial Report:** Within 4 hours of detection
- **Final Report:** Within 72 hours
- **Recipient:** NBS (Narodna Banka Srbije)

Incident Types

1. **P0: Data Breach** (PII leaked)
2. **P1: Service Outage** (> 4 hours)
3. **P2: Unauthorized Access** (admin account compromised)
4. **P3: Payment Fraud** (transaction manipulation)

Response Steps

Phase 1: Detection (T+0)

1. **Identify incident:**

- Sentry alert: "Unhandled exception: Unauthorized access"
- Audit log: Multiple failed login attempts from same IP
- User report: "I didn't authorize this transaction"

2. **Classify severity:**

- P0: Data breach (PII leaked)
- P1: Service outage (> 4 hours)
- P2: Unauthorized access (admin account compromised)
- P3: Payment fraud (transaction manipulation)

Phase 2: Containment (T+15 min)

1. **Isolate affected systems:**

- If admin account compromised: Revoke JWT tokens
- If database breach: Block external IPs in NSG
- If payment fraud: Suspend affected user accounts

2. **Preserve evidence:**

- Export audit_log (last 7 days)
- Export PostgreSQL WAL logs
- Screenshot Sentry errors

3. **Notify CEO (Alem Basic):**

- Slack DM + SMS + Email
- Include: Incident type, affected users, containment status

Phase 3: Eradication (T+1 hour)

1. **Fix root cause:**

- If SQL injection: Patch vulnerable endpoint
- If leaked credentials: Rotate secrets (Key Vault)
- If DDoS: Enable Azure WAF rate limiting

2. **Deploy fix:**

- Create hotfix branch
- Deploy via `deploy-production.yml` (fast-track, skip canary)

3. **Verify fix:**

- Penetration test (Securion)
- Audit log review (no new suspicious activity)

Phase 4: Recovery (T+2 hours)

1. **Restore service:**

- Re-enable affected user accounts
- Disable maintenance mode

2. **Notify affected users:**

- Email: "Security incident resolved, your account is safe"
- In-app notification

3. **Monitor for recurrence:**

- Grafana: Watch error rate dashboard
- Sentry: Check for similar errors

Phase 5: Reporting (T+4 hours)

1. **NBS Initial Report** (within 4 hours):

- **Template:** docs/compliance/nbs-incident-report-template.md
- **Fields:** Incident type, timestamp, affected users, containment status
- **Submit:** Email to nbs@nbs.rs + online portal (if exists)

2. **Poverenik Initial Report** (within 72 hours if PII breach):

- **Template:** docs/compliance/poverenik-incident-report-template.md
- **Fields:** Data categories affected, number of users, mitigation steps
- **Submit:** Email to office@poverenik.rs

Phase 6: Final Report (T+72 hours)

1. **NBS Final Report:**

- Root cause analysis
- Timeline of events
- Mitigation measures implemented
- Lessons learned

2. **Internal Post-Mortem:**

- Blameless review (CodeCraft + Securion + John)
- Action items (MC tasks)
- Update runbooks

Evidence: All incidents logged in docs/incidents/ (YYYY-MM-DD-incident-name.md)

Runbook 4: Release & Rollback

Release (< 5 min)

Trigger: Git tag v1.1.0 pushed to main

Automated Steps (deploy-production.yml):

1. Build Docker images (backend + frontend)
2. Push to ACR (tag: v1.1.0 + latest)
3. Deploy to "green" revision (Azure Container Apps)

4. Health check green revision (`GET /health`)
5. Route 10% traffic to green (canary)
6. Wait 5 minutes (monitor error rate)
7. If error rate < 1%: Route 100% traffic to green
8. If error rate $\geq$ 1%: Rollback to blue (see below)

Manual Verification:

```
# Check green revision health
curl https://green--dropsrbija-backend.azurecontainerapps.io/health

# Check error rate (Grafana)
open https://grafana.drop.rs/d/errors

# Check Sentry (last 5 min)
open https://sentry.io/organizations/alai/issues/
```

Rollback (< 5 min)

Trigger: Error rate $\geq$ 1% during canary phase OR manual decision

Steps:

1. Route 100% traffic to blue revision (previous stable):

```
az containerapp revision set-mode \
  --name dropsrbija-backend \
  --resource-group dropsrbija-prod \
  --mode single \
  --revision dropsrbija-backend--v1.0.0 # Previous stable
```

2. Deactivate green revision:

```
az containerapp revision deactivate \
  --name dropsrbija-backend \
  --resource-group dropsrbija-prod \
  --revision dropsrbija-backend--v1.1.0 # Failed release
```

3. Verify rollback:

```
curl https://drop.rs/health
# Should return version: "1.0.0" (previous stable)
```

4. Notify team (Slack #drop-srbija):

- "Rollback complete: v1.1.0 → v1.0.0"
- "Investigating root cause, will retry deployment after fix"

RTO: < 5 minutes (no rebuild required, revision swap only)

Last Updated: 2026-04-17

Maintained By: FlowForge (Kelsey Hightower)

Revision #1

Created 2026-07-29 00:36:17 UTC by John

Updated 2026-07-29 00:36:17 UTC by John