Incident Report Template

Incident Report

> **Project:** Bilko **Version:** 0.1 **Date:** 2026-02-23 **Author:** Ops Architect **Status:** Draft (Template — fill in per incident) **Reviewers:** Tech Lead, Alem Bašić

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Ops Architect	Initial draft

INSTRUCTIONS

Create a new incident report file for each incident:

- Filename: `INCIDENT-YYYY-MM-DD-<short-title>.md`
- Location: `docs/operations/incidents/`
- Fill in all sections within 48 hours of incident resolution
- P0 incidents require a full post-mortem (see `post-mortem.md`)

Incident Report: [SHORT TITLE]

Incident ID: INC-YYYY-MM-DD-NNN **Reported by:** [Name] **Date detected:** YYYY-MM-DD HH:MM CET **Date resolved:** YYYY-MM-DD HH:MM CET **Total duration:** X hours Y minutes **Severity:** P0 / P1 / P2 / P3

---

1. Incident Summary

[Example: On YYYY-MM-DD at HH:MM CET, the Bilko API became unavailable due to an out-of-memory error on Railway. All users were unable to create invoices or access their dashboard for 47 minutes. The incident was resolved by restarting the Railway service and deploying a memory optimization patch.]

---

2. Impact

Metric	Value
Duration	X min
Users affected	[All / Specific org / None]
Data loss	[None / Describe if any]
Financial records at risk	[None / Describe if any]
Revenue impact	[None / Describe if applicable]
GDPR reportable	[Yes / No] — if Yes, notify Datatilsynet within 72h

---

3. Timeline

Time (CET)	Event
HH:MM	First signs of degradation (detected by: BetterStack / user report / Sentry)
HH:MM	Incident declared by [Name]
HH:MM	[First diagnosis step]
HH:MM	[Root cause identified]
HH:MM	[Fix applied]
HH:MM	Service restored, health check green
HH:MM	Incident resolved, monitoring period started
HH:MM	Monitoring period ended — all clear

---

4. Root Cause Analysis

Root cause: [One sentence — the actual technical cause]

Example root causes for Bilko:

- Railway API service exhausted 2GB RAM limit due to memory leak in invoice PDF generation
- Database connection pool exhausted (25 max connections on Railway Starter) under load
- Bad database migration caused index corruption on `invoices` table
- Expired SendGrid API key (not rotated) caused all invoice emails to fail
- Cloudflare R2 API credentials rotated without updating Railway env vars

Contributing factors:

- [Factor 1: e.g., No memory usage alerting configured]
- [Factor 2: e.g., No automated secret rotation reminder]

---

5. Detection

How was the incident detected?

- ☐ BetterStack uptime monitor alert
- ☐ Sentry error rate spike
- ☐ User report (direct to Alem / support)
- ☐ Routine health check
- ☐ Monitoring dashboard review

Time to detect: [X minutes from first symptom to alert]

Was detection fast enough? [Yes / No — if No, document what would have caught it faster]

---

6. Response

Response Actions Taken

Time	Action	Result
HH:MM	[Action taken]	[Result]

What Worked Well

- [e.g., BetterStack alert fired within 2 min]
- [e.g., Rollback procedure was documented and worked on first try]

What Didn't Work

- [e.g., Railway logs were hard to search for the specific error]
- [e.g., No backup Railway contact — only one person on-call]

---

7. Financial Data Integrity Check

Required for all P0/P1 incidents. Skip for P2/P3 that don't touch accounting.

- ☐ All invoices created during incident window verified (count before vs after)
- ☐ VAT calculations for affected period verified correct
- ☐ Double-entry balances verified for all affected transactions
- ☐ No orphaned transactions (debit without credit) in database
- ☐ Affected organization(s) notified if any data discrepancy found

Verification query (run after incident):

```
-- Check for unbalanced entries during incident window
SELECT t.id, t.created_at, t.amount
FROM transactions t
WHERE t.created_at BETWEEN '<incident_start>' AND '<incident_end>'
  AND t.id NOT IN (
    SELECT DISTINCT "transactionId" FROM transaction_entries WHERE type = 'debit'
  );
-- Expected: 0 rows (all transactions have debit entries)

-- Check invoice totals match line item sums
SELECT i.id, i.total_amount,
       SUM(ii.quantity * ii.unit_price * (1 + ii.tax_rate/100)) as calculated_total
FROM invoices i
JOIN invoice_items ii ON ii."invoiceId" = i.id
WHERE i.created_at BETWEEN '<incident_start>' AND '<incident_end>'
GROUP BY i.id, i.total_amount
HAVING ABS(i.total_amount - SUM(ii.quantity * ii.unit_price * (1 + ii.tax_rate/100))) >
0.0001;
```

```
-- Expected: 0 rows
```

8. User Communication

Channel	When Sent	Content Summary
status.bilko.io	HH:MM	"Investigating service disruption"
status.bilko.io	HH:MM	Status update with ETA
status.bilko.io	HH:MM	"Service restored"
Email to affected users	HH:MM (if needed)	[Summary of impact and resolution]

9. Action Items

#	Action	Owner	Due Date	Priority
1	[Preventive action]	[Owner]	YYYY-MM-DD	P0/P1/P2
2	[Detection improvement]	[Owner]	YYYY-MM-DD	P1
3	[Documentation update]	[Owner]	YYYY-MM-DD	P2

10. Follow-Up

- ☐ Post-mortem scheduled (required for P0 incidents): [Date/Time]
- ☐ Action items added to project backlog
- ☐ Runbook updated with new diagnosis/fix steps
- ☐ Monitoring improved to detect this issue faster next time

Approval

Role	Name	Date	Signature
Author (on-call)			
Reviewer	Alem Bašić		

Revision #3
Created 2026-02-24 23:11:22 UTC by John
Updated 2026-05-31 20:04:07 UTC by John