Credential Rotation Guide

Credential Rotation — Consolidated Guide

Last Updated: 2026-02-10 **Status:** Infrastructure Complete, Awaiting Execution **Original Docs:** Consolidated from 8 separate files (2026-01-31)

Overview

Comprehensive credential rotation system for BasicAS Group. Handles API keys, tokens, and service credentials across 5+ platforms.

Key Principle: Manual collection (30 min) → Automated storage (2 min) → Secure keychain

---

Why Manual Collection is Required

Technical Reality:

1. All services require authenticated browser login
2. No programmatic API exists to regenerate credentials without authentication
3. Alem's login credentials not stored in system (by design)
4. MFA/CAPTCHA may be required at some services
5. OAuth flows require interactive user approval

Result: Automation handles storage and verification. Human handles credential generation.

Services Requiring Rotation

Service	Type	Location	Rotation Frequency

Anthropic	API Key	console.anthropic.com/account/keys	90 days
ElevenLabs	API Key	elevenlabs.io/app/profile/api-keys	90 days
Telegram	Bot Token	@BotFather in Telegram app	As needed
Discord	Bot Token	discord.com/developers/applications	As needed
Z.ai	API Key	z.ai dashboard	90 days
one.com	SMTP Password	one.com webmail settings	180 days
Cloudflare	API Token	dash.cloudflare.com/profile/api-tokens	90 days

---

Current Credential Status (as of 2026-01-31)

? Found in System

- **Anthropic:** `sk-ant-api03-cH6oDXQ...` (in config files)
- **ElevenLabs:** `sk_5b54ee283cd9ee9ff12dae6...` (in config files)
- **Telegram:** `8390640424:AAFP9Zf1R9vUV6T7aQhW6Ow7Z3-13aGDw1g` (in config files)
- **Z.ai:** `143eda33ea3e41f89df91e0e6fd1f0bb.VZxsj7NcstKy9UWX` (in config files)

? Not Found

- **Discord:** Needs creation (bot not yet set up)

Rotation Process

Step 1: Review Services (5 min)

Identify which credentials need rotation based on:

- Last rotation date
- Security incidents

- Compliance requirements
- Service recommendations

Step 2: Collect New Credentials (25-30 min)

Visit each service and generate new credentials:

Anthropic (5 min)

1. Visit https://console.anthropic.com/account/keys
2. Login with Alem's credentials
3. Click "Create Key"
4. Name: `BasicAS-Production-YYYY-MM-DD`
5. Copy key (starts with `sk-ant-api03-`)
6. Save to temp file

ElevenLabs (5 min)

1. Visit https://elevenlabs.io/app/profile/api-keys
2. Login with Alem's credentials
3. Click "Generate API Key"
4. Copy key (starts with `sk_`)
5. Save to temp file

Telegram Bot (5 min)

1. Open Telegram app
2. Message @BotFather
3. Command: `/mybots`
4. Select bot → API Token → Regenerate Token
5. Copy token (format: `NNNNN:AAAAAAAA...`)
6. Save to temp file

Discord Bot (5 min)

1. Visit https://discord.com/developers/applications
2. Login with Alem's credentials
3. Select application → Bot → Reset Token
4. Copy token
5. Save to temp file

Z.ai (5-10 min)

1. Visit https://z.ai
2. Login with Alem's credentials
3. Navigate to API settings

4. Generate new token
5. Copy token
6. Save to temp file

Step 3: Create Credential File (2 min)

Format: `~/temp-credentials.txt`

```
ANTHROPIC_API_KEY=sk-ant-api03-...
ELEVENLABS_API_KEY=sk_...
TELEGRAM_BOT_TOKEN=1234567890:AAAAA...
DISCORD_BOT_TOKEN=...
ZAI_API_KEY=...
```

Step 4: Run Import Tool (2 min)

```
# Import to macOS Keychain + update configs
node ~/system/tools/credential-import.js ~/temp-credentials.txt

# Verify storage
security find-generic-password -s "ANTHROPIC_API_KEY"
security find-generic-password -s "ELEVENLABS_API_KEY"
```

Step 5: Update Service Configs (5 min)

The import tool auto-updates:

- `~/system/config/*.json` files
- Environment variables
- Docker compose files (if applicable)

Manual verification required for:

- Any hardcoded credentials in code
- Third-party integrations
- CI/CD pipelines

Step 6: Test Services (10 min)

```
# Test Anthropic
curl -H "x-api-key: $ANTHROPIC_API_KEY" https://api.anthropic.com/v1/models

# Test ElevenLabs
curl -H "xi-api-key: $ELEVENLABS_API_KEY" https://api.elevenlabs.io/v1/user

# Test Telegram Bot
curl https://api.telegram.org/bot$TELEGRAM_BOT_TOKEN/getMe
```

Step 7: Revoke Old Credentials (5 min)

Go back to each service and **delete/revoke old credentials** to prevent unauthorized use.

Automation Tool: credential-import.js

Location: ~/system/tools/credential-import.js (500+ lines) **Status:** □ READY TO USE (built 2026-01-31)

Features:

- Reads credentials from text file
- Saves all to macOS Keychain (encrypted)
- Updates ~/system/config/ JSON files
- Generates summary report
- Securely deletes temp files (shred + rm)

Usage:

```
node ~/system/tools/credential-import.js <credentials-file>
```

Output:

- Keychain entries created
- Config files updated
- Summary report printed
- Temp file securely deleted

Security Notes

? Secure Practices

- Credentials stored in macOS Keychain (encrypted at rest)
- Temp files securely deleted after import (7-pass shred)
- API keys use principle of least privilege
- Rotation schedule enforced (90-180 days)

? Current Weaknesses

- Some credentials still in plaintext in docker-compose.yml files (task #310)
- No automated expiry monitoring
- No central credential vault (consider Vault or 1Password)

? Recommended Improvements

1. **Move Docker secrets:** Extract passwords from docker-compose.yml to Docker secrets
2. **Automated expiry alerts:** Script to check credential age and send alerts
3. **Central vault:** Migrate to HashiCorp Vault or 1Password for centralized management
4. **Audit logging:** Track who accessed which credential when

---

Compliance & Best Practices

Rotation Schedule

- **High-risk credentials (admin, root):** 30 days
- **API keys (production):** 90 days
- **Service passwords (SMTP, DB):** 180 days
- **Personal accounts:** Annually

Documentation Requirements

- Record rotation date in CHANGELOG
- Update this document with new status
- Notify team of service disruptions

Incident Response

If credential compromise suspected:

1. **Immediately rotate** affected credential
2. Audit logs for unauthorized usage
3. Revoke old credential
4. Document incident in ~/system/reports/security/
5. Review access policies

Git Remote Embedded Credentials (added 2026-07-29, MC #99653)

Distinct failure mode from the API-key rotation above: a token embedded directly in a git remote URL (`https://user:TOKEN@github.com/...`), stored in plaintext in `.git/config`. This file is never tracked/committed but is trivially readable by anyone with local filesystem access, and can leak via backups, `tar`/zip of a project directory, `.git` copied into a support ticket, etc.

Detection

```
grep -n 'github_pat_\|ghp_\|gho_\|ghu_\|ghs_\|ghr_' <repo>/.git/config
# Repo-wide sweep:
grep -l 'github_pat_\|ghp_' ~/projects/*/.git/config ~/companies/*/.git/config ~/*/.git/config
\
  ~/business/*/.git/config ~/business/*/*/.git/config ~/clients-external/*/.git/config
2>/dev/null
```

Remediation

1. **Revoke the token at the source** — github.com/settings/tokens (fine-grained or classic). This requires an authenticated human session or a valid `gh auth login`; there is no unauthenticated API path to revoke someone else's-visible token.
2. Switch the remote to one of:
   - SSH: `git remote set-url origin git@github.com:<owner>/<repo>.git`
   - HTTPS + OS credential helper: `git config credential.helper osxkeychain` (macOS), then `git remote set-url origin https://github.com/<owner>/<repo>.git` (no embedded secret).
3. Re-run the detection grep to confirm the working tree's `.git/config` is clean.
4. Check `.git/config` is the ONLY place a URL-embedded credential can hide — reflog/packed-refs do not carry remote URLs, but old shell history, CI logs, or cloned copies elsewhere might.

Incident log

- MC #99653 (found 2026-05-07, closed-partial 2026-07-29): `~/system/.git/config` origin URL for `johnatbasicas/clawd` had an embedded `github_pat_11B5LV...`. By the time of remediation the working config was already clean (no embedded token) — likely as a side effect of the 2026-07-15 git-bomb filter-repo rewrite (MC #105751), though no direct evidence ties the two. Actual GitHub-side token revocation could not be verified/performed by an automated agent session (no valid `gh auth`, no vault access) — requires manual CEO action at github.com/settings/tokens. See `~/system/evidence/99653/verdict.md`.

Related Documents

Original Files (Archived)

- `CREDENTIAL-ROTATION-2026-01-31.md` - Initial audit
- `CREDENTIAL-ROTATION-ACTION-PLAN.md` - Step-by-step guide
- `CREDENTIAL-ROTATION-COMPLETE.md` - Infrastructure completion report
- `CREDENTIAL-ROTATION-FINDINGS.md` - Security findings
- `CREDENTIAL-ROTATION-INDEX.md` - File index
- `CREDENTIAL-ROTATION-README.md` - Quick reference
- `CREDENTIAL-ROTATION-STATUS.md` - Technical status
- `CREDENTIALS-ROTATION-README.md` - User guide
- `SUBAGENT-CREDENTIAL-ROTATION-REPORT.md` - Subagent execution report
- `SUBAGENT-CREDENTIAL-ROTATION-STATUS.md` - Subagent status

All originals preserved in: ~/system/context/docs/security/ (timestamped)

Next Steps

1. **Schedule rotation:** Add to calendar (quarterly for API keys)
2. **Execute rotation:** Follow Step 1-7 above
3. **Document completion:** Update this file with new dates
4. **Improve automation:** Build expiry monitoring script

Maintained by: John (AI Director) **Reviewed by:** Alem (CEO) **Next Review:** 2026-05-10 (90 days)

Revision #7
Created 2026-02-17 22:14:37 UTC by John
Updated 2026-07-29 01:20:34 UTC by John