Sovereign Sanctuary Elite - Safety Protocol

Version: 2.0.0 **Date:** 2026-01-25 **Classification:** Operational Security Documentation

---

1. Overview

This document defines the safety protocols and failsafe mechanisms built into the Sovereign Sanctuary Elite system. All operations are designed to be **safe by default**, with explicit confirmation required for any potentially destructive action.

2. Core Safety Invariants

The system enforces these invariants at all times:

Invariant	Value	Description
`status`	GREEN	System must be in healthy state
`readiness`	100	All components must be ready
`governance_mode`	ACTIVE	Governance controls must be enabled

If any invariant is violated, all operations are blocked.

3. Safety Tools

3.1 Safety Guardrail Check

Location: `tools/safety/safety_guardrail_check.py`

Purpose: Hard fail if system invariants are violated. This script MUST pass before any snapshot, restore, or takedown operation.

Usage:

```
python tools/safety/safety_guardrail_check.py
```

Exit Codes:

- `0` - All guardrails passed, safe to proceed
- `1` - Guardrails failed, operations blocked

No override flag exists. If this fails, the system is not safe to operate.

3.2 Snapshot Safety Verifier

Location: `tools/safety/verify_snapshot_safety.py`

Purpose: Verify integrity and temporal consistency of evidence snapshots.

Usage:

```
python tools/safety/verify_snapshot_safety.py EVIDENCE_SNAPSHOTS/SNAPSHOT_*
```

3.3 Restore Point Verifier

Location: `tools/safety/verify_restore_point.py`

Purpose: Verify integrity of restore points before restoration.

Usage:

```
python tools/safety/verify_restore_point.py RESTORE_POINTS/RESTORE_*
```

4. Restore Point System

4.1 Creating Restore Points

Location: `tools/restore/create_restore_point.py`

Purpose: Create immutable restore points with cryptographic manifests.

Safety Guarantees:

- Never overwrites existing restore points
- All files hashed with SHA-256
- Only files in allowlist are included
- Manifest is cryptographically sealed

Usage:

```
python tools/restore/create_restore_point.py
```

4.2 Restoring from Points

Location: `tools/restore/restore_from_point.py`

Purpose: Restore system state from a verified restore point.

Safety Guarantees:

- Requires explicit `--confirm` flag
- Creates backup before restore (unless `--skip-backup`)
- Verifies restore point integrity first
- All actions logged to ledger

Usage:

```
# Dry run (default)
python tools/restore/restore_from_point.py RESTORE_POINTS/RESTORE_*

# Execute restore
python tools/restore/restore_from_point.py RESTORE_POINTS/RESTORE_* --
confirm
```

5. Evidence Snapshot System

5.1 Creating Snapshots

Location: `tools/snapshot/create_evidence_snapshot.py`

Purpose: Create cryptographically sealed evidence for external verification.

Contents:

- Selected evidence files
- MANIFEST.json with SHA-256 hashes
- VERIFY.sh for one-command verification
- README.md for external parties

Usage:

```
python tools/snapshot/create_evidence_snapshot.py
```

6. Mirror Takedown Protocol

6.1 Takedown Script

Location: `tools/takedown/mirror_takedown.py`

Purpose: Controlled takedown of mirror deployments.

Safety Guarantees:

- **DEFAULT IS DRY RUN** - No changes without `--execute`
- Requires hash-matched confirmation
- Manual steps enforced for destructive operations
- All actions logged to ledger

Supported Modes:

- `local` - Local mirror directories
- `github` - GitHub repository
- `ipfs` - IPFS pinned content
- `s3` - S3 bucket
- `all` - All of the above

Usage:

```
# Get state hash
sha256sum runtime/state.json

# Dry run (default)
python tools/takedown/mirror_takedown.py --mode local --confirm-hash <HASH>

# Execute (shows manual steps)
python tools/takedown/mirror_takedown.py --mode local --confirm-hash <HASH>
--execute
```

7. Standard Operating Procedures

7.1 Safe Run Order

Always execute in this order:

```
# 1. Check safety guardrails
python tools/safety/safety_guardrail_check.py

# 2. Create restore point (before any changes)
python tools/restore/create_restore_point.py

# 3. Perform operations...

# 4. Create evidence snapshot
python tools/snapshot/create_evidence_snapshot.py

# 5. Verify snapshot
python tools/safety/verify_snapshot_safety.py EVIDENCE_SNAPSHOTS/SNAPSHOT_*
```

7.2 Emergency Rollback

If something goes wrong:

```
# 1. Verify restore point integrity
python tools/safety/verify_restore_point.py RESTORE_POINTS/RESTORE_*

# 2. Execute restore
python tools/restore/restore_from_point.py RESTORE_POINTS/RESTORE_* --
confirm
```

8. Security Guarantees

Guarantee	Implementation
Deterministic file ordering	Sorted paths in manifests
Hash-anchored authorization	SHA-256 confirmation required
No silent deletes	All deletions require explicit flags
Restore points immutable	Never overwritten, only created
External proof separated	Snapshots vs restore points
IDE-safe	No shell magic required

9. Audit Trail

All safety-related operations are logged to `evidence/ledger.jsonl`:

- `GUARDRAIL_CHECK` - Safety check results
- `RESTORE_POINT_CREATED` - New restore point
- `RESTORE_EXECUTED` - Restore operation
- `EVIDENCE_SNAPSHOT_CREATED` - New snapshot
- `MIRROR_TAKEDOWN` - Takedown operation

This system is failsafe by construction.

If you encounter a situation not covered by this protocol, **stop and assess** before proceeding.