ChatGPT

Hash Artifact Manifest and Validation Guide

Purpose

This document explains the origin, location, and purpose of all SHA-256 hash artifacts and signatures generated in the Micron submission pipeline. It covers the interaction between multiple Makefiles and scripts across the `micron-casefile` repository and its `submission-packet` submodule (formerly `evidence-packet`).

The system implements cryptographic integrity measures to ensure: - File-level immutability - Submission authenticity - Auditability of the final deliverables

---

Hash Artifact Summary

All hash outputs are stored centrally in:

```
/home/sef/Repos/micron_case_git_workspace/micron-casefile/hashes
```

Files Generated

File	Description
`exhibits_sha256_hashes.txt`	SHA-256 digests of all files in `submission-packet/`
`exhibits_sha256_hashes.txt.asc`	GPG signature of the above, created with `18759C0DBE1B112F`
`repo_sha256_hashes.txt`	Summary of top-level repo commit and submodule SHAs
`repo_sha256_hashes.txt.asc`	GPG signature of the above

🕑 Completed on: `Fri Jun 6 04:21:10 PM MDT 2025`

Generation and Signing Workflow

`submission-packet/Makefile`

From inside the submodule:

```
make hash   # SHA-256 of submission-packet contents
make sign   # GPG-sign the hash file
make verify # Validate GPG signature integrity
```

Each `make` command prints detailed messages and writes output to:

```
../hashes/
```

This ensures the `micron-casefile` parent repo always holds authoritative SHA snapshots.

Top-Level Generation (`micron-casefile`)

```
make gen    # Runs gen-sha256-top-level.sh
make verify # Validates top-level signature
make clean  # Clears out hash artifacts
make sync   # Runs update-everything.sh end-to-end
```

These commands: - Record HEAD SHA of the container repo - Capture submodule commit SHAs - Sign it using GPG

Why Store Everything in `micron-casefile/hashes/`?

1. **Canonical Location**: All SHA metadata—regardless of source—lives under one root.
2. **Signature Consolidation**: Only one GPG key is needed to sign everything.
3. **Submission Integrity**: `submission-packet` can be evaluated from the parent repo without needing internal logic.
4. **Audit Trail**: Any third party can verify hashes without modifying the `submission-packet` artifact.

Provenance and Trust

GPG Key Used: `18759C0DBE1B112F`

Sample signature validation:

```
gpg --verify hashes/exhibits_sha256_hashes.txt.asc
```

Output:

```
gpg: Good signature from "Seaph Antelmi <seaph@wolfmind.ai>"
```

Final Notes

- The SHA tools are deterministic. Re-running them on unchanged content yields identical outputs.
- All Makefiles and scripts have been standardized to emit helpful, readable, colorized logs.
- The central `hashes/` directory will eventually form part of the deliverable audit bundle.

Powered by Wolfmind

Where ethical architectures meet provable integrity.