SAGE Implementation Guide

How to Use the SAGE Specification, Template, and Tooling

Cloud Security Alliance AI Safety Initiative

April 2026 | Version 1.0-Draft

Companion to SAGE Specification v1.0-Draft

Table of Contents

Note: Update this table of contents after opening in Word (right-click and select Update Field).

What This Package Contains

This implementation guide accompanies two other documents. Together, the three form a complete package for publishing and consuming SAGE-formatted security research.

Document	Purpose
SAGE Specification v1.0-Draft	The normative spec. Defines every frontmatter field, controlled vocabulary, content hash computation, conformance criteria, and JSON Schema. This is what parsers and validators implement against.
SAGE Document Template	A blank, commented markdown file. Fill in the fields, write your content in the body sections, compute the content hash, and publish. Every field includes inline guidance explaining what it is and when you need it.
This Guide	The document you are reading. Explains how the pieces fit together, walks through the authoring workflow, and answers the questions you will have the first time you create an SAGE document.

The Problem in 30 Seconds

Security research is published in formats designed for human readers. The fastest-growing category of consumers is not human. It is AI-powered tools: RAG pipelines, security copilots, and autonomous agents that ingest, index, and retrieve security knowledge at scale.

Existing machine-readable standards (STIX, OSCAL, CycloneDX, SARIF) handle structured data objects well. None of them carries the analytical reasoning, implementation guidance, and contextual narrative that makes up the bulk of security publications. This is the structured narrative gap.

SAGE fills that gap. It is a CommonMark plus YAML frontmatter convention that provides a machine-native distribution channel alongside human-optimized formats like PDF. The same source feeds both channels.

Authoring Workflow: Start to Finish

Creating an SAGE document takes six steps. The template handles steps 1 through 3 for you.

Step 1: Open the template

Open SAGE-Document-Template-v1.0.md in any text editor or markdown editor. The file contains every possible frontmatter field with inline comments explaining what each one does, when it is required, and what values it accepts.

Step 2: Fill in the frontmatter

Start with the required fields. Every SAGE document must have these ten fields filled in:

Field	What to Enter	Example
title	Your document's title	Shadow AI Risk Assessment
document_id	{ORG}-{YEAR}-{TYPE}-{SEQ}	CSA-2026-TA-003
version	Semantic version or increment	1.0.0
date	ISO 8601 date	2026-04-01
status	draft, review, final, or superseded	final
document_type	One of seven categories	threat_analysis
content_domain	One or more knowledge domains	["ai_security"]
authors	Author names or organizations	["J. Smith"]
organization	Publishing organization	Cloud Security Alliance
generation_metadata.authored_by	human, ai, or human_ai_collaborative	human

If your document involved AI assistance (authored_by is "ai" or "human_ai_collaborative"), you must also fill in model_id, model_version, human_review, and review_attestation. These fields are conditional: required when the condition applies, ignored otherwise.

Then add whichever optional fields are relevant. The template includes them all with guidance. Delete the ones you do not use.

Step 3: Write your content

Below the closing --- delimiter, write your content using standard markdown. Follow the section structure in the template:

H1 title: Must match the title field in frontmatter. Appears exactly once.

Abstract (H2): A self-contained summary. RAG systems often retrieve only this section to decide whether to fetch the full document.

Body sections (H2/H3): Each H2 section should be independently meaningful when extracted as a chunk. Do not skip heading levels.

Control mapping table: Optional. Use a standard markdown table with framework identifiers as column headers. Parsers will detect and extract it automatically.

References (H2): Required when you use in-text citations. Use numbered brackets [1], [2] in the body, and provide full entries in this section.

Step 4: Add data markings

If your document has handling restrictions, add the tlp field to the frontmatter. SAGE uses FIRST TLP 2.0 designations: TLP:RED, TLP:AMBER, TLP:AMBER+STRICT, TLP:GREEN, or TLP:CLEAR. If you omit it, consumers treat the document as TLP:CLEAR.

This matters for agentic workflows. When AI agents consume SAGE documents, they must respect TLP markings. Agents must not include content from TLP:RED or TLP:AMBER+STRICT documents in context windows shared with other agents. Derivative documents must carry a TLP at least as restrictive as their sources.

Step 5: Compute the content hash

Do this last, after all editing is complete. The content_hash field contains a SHA-256 hash of everything below the closing --- delimiter. The computation has precise rules to ensure two implementations produce the same hash for the same document:

```
1. Take all bytes after the closing ---\n delimiter
2. Replace all CR+LF and standalone CR with LF
3. Apply Unicode NFC normalization
4. Compute SHA-256
5. Encode as 64-character lowercase hex
```

Tooling will automate this. Until tooling exists, you can compute it with a command-line one-liner or a short script. The spec includes test vectors in Appendix B so you can verify your implementation.

Step 6: Publish

Publish the .md file as your machine-native distribution channel. Continue producing PDF or HTML for human readers from the same source. The double conversion disappears because the publication format and the consumption format are the same.

Document Identifiers: How to Avoid Collisions

Every SAGE document needs a globally unique document_id. The spec defines a namespace convention to prevent collisions across publishers:

```
{ORG}-{YEAR}-{TYPE}-{SEQ}
```

ORG is your organization's uppercase abbreviation. You pick it. YEAR is four digits. TYPE is a two-to-four character code matching your document_type (WP for whitepaper, TA for threat_analysis, GD for guidance, and so on). SEQ is a zero-padded sequence number you manage internally.

CSA and OWASP are reserved prefixes for those organizations. Everyone else self-assigns. If you are ACME Corp, use ACME. If you are an independent researcher, use your initials or a unique abbreviation.

Cross-references between SAGE documents use the format document_id#section-slug for section-level precision. Example: CSA-2026-WP-042#control-mapping-table.

Integrity: Why Content Hashing Matters

SAGE documents are designed as RAG pipeline inputs. Structured documents with consistent metadata get preferential retrieval treatment in vector search implementations. Research on RAG corpus poisoning has demonstrated attack success rates exceeding 90% with as few as five injected texts.

A well-crafted poisoned SAGE document with legitimate-looking frontmatter is a more effective attack vector than unstructured injection because the format's own metadata makes it look authoritative. The content_hash field provides tamper detection. The optional signature block enables publisher authentication.

When a consumer ingests an SAGE document, it can recompute the hash and compare it to the frontmatter value. If they do not match, the document has been modified since publication. This is a simple, lightweight integrity check that catches both deliberate tampering and accidental corruption from extraction pipelines.

Data Marking and Trust Boundaries

SAGE integrates FIRST TLP 2.0 designations directly into the frontmatter. This is critical for enterprise environments where security research crosses trust boundaries, and essential for agentic workflows where agents assemble context from multiple sources.

TLP Designation	Handling Restriction
TLP:RED	For named recipients only. No further disclosure.
TLP:AMBER	Limited disclosure within recipient's organization and clients.
TLP:AMBER+STRICT	Limited disclosure within recipient's organization only.
TLP:GREEN	Limited disclosure within the recipient's community.
TLP:CLEAR	No restrictions on disclosure.

When agents generate derivative SAGE documents that incorporate content from marked sources, the derivative must carry a TLP designation at least as restrictive as the most restrictive source. The data_marking field provides a free-text slot for handling restrictions beyond TLP (e.g., "INTERNAL USE ONLY").

For Consumers: What to Do with SAGE Documents

If you build or operate tools that ingest security research, SAGE documents give you structured metadata you can act on immediately.

RAG pipelines

Use the recommended_chunk_level field to choose your chunking strategy. Use abstract_for_rag for two-stage retrieval: screen documents by abstract before retrieving full content. Use content_domain and frameworks_referenced to filter retrieval to relevant domains.

Security platforms (SIEM, SOAR, GRC, CNAPP)

Extract YAML frontmatter to populate knowledge base metadata. Use controls_mapped to link narrative guidance to specific control identifiers in your catalog. Parse control mapping tables to automate cross-framework referencing.

Agentic workflows

Check content_hash before trusting the document. Respect tlp markings when assembling context across agents. Use the related_documents array to traverse the knowledge graph and discover connected analysis.

Conformance: How to Know You Got It Right

The spec defines conformance for both documents and parsers.

A conforming SAGE document

Has valid YAML frontmatter with all required fields present and correctly typed. Uses values from the controlled vocabularies (or valid extensions for extensible fields). Contains a valid content_hash. Follows the heading hierarchy conventions in the body.

A conforming SAGE parser

Extracts and validates frontmatter. Accepts unrecognized values in extensible fields without error. Identifies control mapping tables. Verifies content hashes (should) and signatures (may). Exposes TLP markings to consuming applications for trust boundary enforcement.

The spec includes a full JSON Schema (Appendix A) that validators can implement against directly.

What to Do Next

Security publishers: Start with one document. Pick a recent whitepaper or guidance document, create the SAGE version using the template, and publish it alongside the existing PDF. The authoring cost is marginal when content originates in structured form.

Tool vendors: Evaluate SAGE as an ingestion format. The JSON Schema in the spec means you can validate frontmatter with existing tooling. The controlled vocabularies mean you can filter and route documents without custom parsing logic.

Standards bodies: The structured narrative gap complements the data object standards you already steward. SAGE is designed for formal standardization, not competition.

Everyone: Read the spec. Try the template. Provide feedback. SAGE is intended to be open, community-governed, and vendor-neutral.

SAGE Specification, Template, and Guide

labs.cloudsecurityalliance.org/sage