



SIEM LOGGING

CUSTOMER OVERVIEW

WHAT THIS IS

Smart Squad emits structured security events: one JSON document for every security-relevant action. Your SIEM platform (Splunk, Microsoft Sentinel, QRadar, ArcSight, Elastic, Wazuh, Sumo Logic, Graylog and others) collects them and correlates them into your existing security operations workflow.

The goal is to give your security team and your compliance auditor a single, machine-readable feed of every security-relevant action taken inside Smart Squad, alongside the rest of the events flowing into your SIEM.

There are two ways to receive the feed. Both carry the same events in the same JSON format, so one set of parsing and detection rules works with either:

DELIVERY MODEL	FOR	HOW THE EVENTS REACH YOUR SIEM
On-premise: Windows Event Log	Smart Squad installed on your agency's own servers	Smart Squad writes each event to the <code>SS_Security</code> Windows Event Log channel. Your SIEM agent or Windows Event Collector picks it up.
Hosted / shared servers: HTTPS push (2.9.5)	Agencies on Smart Squad servers they do not operate, including shared multi-agency hosting	Smart Squad sends your agency's events, and only your agency's events, to an HTTPS endpoint your SIEM provides. No agent is needed on the Smart Squad servers.

WHAT GETS CAPTURED

CATEGORY	EXAMPLES
Authentication	Login success, login failure, account lockout, MFA challenge, MFA success/failure, password reset request, password reset completion, logout
Data access (reads)	Note viewed, report viewed, ticket viewed, person queried, vehicle queried, intel posting viewed, face search performed, every Smart Squad search (Search Audit), scan hand-off collected by the in-car device (2.9.5)
Data modification (writes)	Notes created and modified (entries, attachments added or removed, exports, review), reports created/saved/exported, tickets created and cancelled, intel posts created/modified, form submissions created, saved, printed or exported, AI activity against a report
Admin actions	User created, enabled or disabled (including automatic disabling of inactive accounts), user profile synchronised from the RMS or Microsoft Entra ID, roles created, changed, disabled or enabled, capabilities granted or revoked, roles assigned to or removed from users, Smart Draft prompt versions changed, system configuration changed in the admin portal

Every event includes:

- **Who:** user ID, login name, agency ID, source IP, user agent (when available from the request)
- **What:** the action and the target record's ID
- **When:** UTC timestamp to millisecond precision
- **Where:** host name and which Smart Squad process emitted the event

- **Which:** a unique event ID (`event.uid`) that identifies this occurrence (2.9.5)
- **Why it failed:** for failures, a structured reason code (e.g. `bad_password` , `invalid_otp` , `max_failed_attempts`)

WHAT IS *NOT* LOGGED

The following are deliberately left out of the SIEM feed:

- **Credentials:** passwords, OTP codes, password reset tokens, raw auth tokens, certificate private material
- **Personally identifiable information (PII) values:** names, dates of birth, plate numbers, addresses, photographs, face embeddings. Only IDs are logged; your SIEM can join back to the source system if your policy permits.
- **Body content:** note bodies, report text, AI prompts and outputs, attachment file content, form field values, face search images
- **Sensitive configuration values:** when a configuration key's name matches a sensitive pattern (`password` , `token` , `secret` , `apikey` , `authtoken` , `privatekey` , `clientsecret` , `connectionstring`), its values are emitted as `[REDACTED]` . Other configuration keys log the old and new values for change tracking.

If your policy requires content auditing, that is the role of the Smart Squad audit tables in the database, which already capture the relevant content context. The SIEM stream is for *event-level*/security telemetry.

THE EVENT FORMAT

Each event is one JSON document. It is identical in both delivery models.

Sample event payload (failed login):

```
{
  "@timestamp": "2026-09-23T14:32:11.482Z",
  "event": {
    "id": 1002,
    "uid": "3f9c2d4e-8a1b-4c7d-9e2f-5a6b7c8d9e0f",
    "name": "auth.login.failure",
    "category": "authentication",
    "severity": "medium",
    "outcome": "failure",
    "reason": "invalid_credentials"
  },
  "actor": {
    "user_id": "1024",
    "username": "jsmith",
    "agency_id": "7",
    "ip": "10.4.12.88",
    "user_agent": "Mozilla/5.0 ..."
  },
  "action": "login",
  "source": {
    "host": "SS-WEB-01",
    "app": "SmartSquad.Service",
    "app_version": "2.9.5.0"
  }
}
```

Sample event payload (note viewed):

```
{
  "@timestamp": "2026-09-23T14:33:01.117Z",
  "event": {
    "id": 3001,
    "uid": "b7e1a0c2-5d3f-4e6a-8b9c-0d1e2f3a4b5c",
    "name": "data.note.viewed",
    "category": "data_read",
    "severity": "info",
    "outcome": "success"
  },
  "actor": {
    "user_id": "1024",
    "username": "jsmith",
    "agency_id": "7",
    "ip": "10.4.12.88"
  },
  "subject": {
    "type": "note",
    "id": "abc123",
    "agency_id": "7"
  },
  "action": "view",
  "source": {
    "host": "SS-WEB-01",
    "app": "SmartSquad.Service",
    "app_version": "2.9.5.0"
  }
}
```

`event.id` identifies the *type* of event (the catalog entry). `event.uid` is unique to each occurrence: use it to remove duplicates.

ON-PREMISE: COLLECTION FROM THE WINDOWS EVENT LOG

Smart Squad writes one JSON document per event to a custom Windows Event Log channel named `SS_Security`. The JSON sits in the event's `Message` field. The event log entry's own `EventID`, `Source` and `Level` are also set, so you can filter at the channel level.

In this model, Smart Squad's responsibility ends at writing to the local Event Log. From there, collection is a standard Windows Event Log integration with your SIEM. There are two common patterns:

Option A: a local agent on each Smart Squad server. Install your SIEM vendor's Windows agent (e.g. Splunk Universal Forwarder, Microsoft Azure Monitor Agent, Elastic Winlogbeat, Wazuh agent, NXLog) and point it at the `SS_Security` channel.

Option B: Windows Event Collector (WEC). Configure each Smart Squad server as a source-initiated subscription that publishes the `SS_Security` channel to a central WEC collector, then run a single SIEM agent on the collector. This reduces agent sprawl when Smart Squad is deployed across several servers.

XPath query for the subscription

```
<QueryList>
  <Query Id="0" Path="SS_Security">
    <Select Path="SS_Security">*</Select>
  </Query>
</QueryList>
```

Prerequisites on each Smart Squad server

Before Smart Squad starts emitting, one-time operator setup is required:

1. **Register the Event Log source.** Run as administrator on each Smart Squad application server:

```
if (-not [System.Diagnostics.EventLog]::SourceExists("SS_Security")) {
    [System.Diagnostics.EventLog]::CreateEventSource("SS_Security", "SS_Security")
}
Limit-EventLog -LogName SS_Security -MaximumSize 512MB -OverflowAction OverwriteAsNeeded
```

Smart Squad will not create the source automatically from a web request; this step must be done separately, by an administrator.

2. **Confirm the channel is visible** in Event Viewer under **Applications and Services Logs > SS_Security**.
3. **Configure your SIEM agent** to read the channel (see Options A and B above).

If the source is missing when Smart Squad starts, it logs a one-time warning to its error trace and does not write events to `ss_security`. The feed "fails open": the application keeps working and only the SIEM feed is affected.

Configuration toggles

Each Smart Squad process (the `SmartSquad.Service` web application and the `SmartSquad.Processor` background service) has SIEM settings in its configuration file (`Web.config` / `App.config`):

SETTING	DEFAULT	PURPOSE
<code>Siem.Enabled</code>	<code>true</code>	Master switch for that process. <code>false</code> stops all SIEM emission from it, in both delivery models. If the setting is missing, emission is off.
<code>Siem.MinSeverity</code>	<code>Info</code>	Drops events below this severity from the <code>ss_security</code> channel. Valid: <code>Info</code> , <code>Low</code> , <code>Medium</code> , <code>High</code> , <code>Critical</code> . HTTPS push uses each agency's own minimum severity instead.
<code>Siem.EventLogSource</code>	<code>SS_Security</code>	Event Log source name. It must match what you registered in step 1 above.
<code>Siem.EventLogChannel</code>	<code>SS_Security</code>	Event Log channel name.
<code>Siem.AppName</code>	(per process)	Identifies which Smart Squad process emitted the event, in <code>source.app</code> .

An on-premise installation that serves several agencies can use a single `ss_security` channel; events are tagged with `actor.agency_id` and `subject.agency_id` for downstream filtering.

HOSTED / SHARED SERVERS: HTTPS PUSH (2.9.5)

When Smart Squad runs on servers your agency does not operate, you cannot install an agent there, and a shared server's Event Log holds every agency's events. In this model Smart Squad instead **pushes your agency's own events to an HTTPS endpoint that your SIEM provides**.

How it works

- When an event occurs, it is queued for your agency in the Smart Squad database. Smart Squad's background service sends queued events to your endpoint over HTTPS, in batches, normally within seconds.
- **Your feed contains only your agency's events.** Each event is assigned to the agency of the user who performed the action (or, for events with no user, the agency that owns the record). Events for other agencies on the same server are never sent to you. Installation-wide maintenance events that belong to no agency are not sent to any agency.
- **Nothing is lost if your endpoint is down.** Events stay queued and are retried at increasing intervals, up to every 15 minutes. If events still cannot be delivered after 72 hours, they are set aside rather than deleted, and Faction Four is alerted. They can be re-sent once the problem is fixed.
- **Delivery is at-least-once.** A retry after a timeout can repeat events your endpoint had already accepted. Remove duplicates on `event.uid`.
- **Order is preserved.** Events are sent in the order they occurred. `@timestamp` is always the time of the event, never the time of delivery.
- **Delivery is monitored.** A rejected credential, an unreachable endpoint or a growing backlog raises an operator alert to Faction Four.

What your agency provides

ITEM	DETAIL
HTTPS endpoint	The URL of an HTTPS input on your SIEM or log pipeline. It must use <code>https</code> , be reachable from the Smart Squad hosting environment, and present a certificate from a publicly trusted certificate authority. On shared servers, endpoints on private or internal network addresses are refused.
Body format	<code>ndjson</code> (newline-delimited JSON) or <code>splunk-hec</code> (Splunk HTTP Event Collector).
Authentication	One of: a bearer token (<code>Authorization: Bearer ...</code>), a custom header and value (e.g. <code>X-API-Key</code> , or <code>Authorization: Splunk <token></code> for Splunk HEC), an HMAC-SHA256 shared secret, or none (for endpoints whose URL itself carries the token).
Firewall	Allow inbound HTTPS from the Smart Squad hosting environment's outbound address(es). Faction Four provides these during onboarding.

Your endpoint URL and credentials are stored encrypted with your agency's data key. Only Faction Four can set or change them; agency administrators cannot redirect the feed.

What Smart Squad sends

- **One HTTPS POST per batch**, of up to 200 events or 1 MB by default.
- **ndjson**: `Content-Type: application/x-ndjson`, one event per line, each line exactly the JSON document shown above.
- **splunk-hec**: `Content-Type: application/json`, one HEC envelope per line: `{"time": <epoch seconds.milliseconds>, "sourcetype": "smartsquad:security", "event": { ... }}`. Point it at your HEC event endpoint (`.../services/collector/event`).
- **Headers on every request**: `X-SmartSquad-Agency` (your agency ID), `X-SmartSquad-Batch-Id` (unique per request) and `X-SmartSquad-Event-Count`.
- **HMAC authentication** adds `X-SmartSquad-Timestamp` (Unix seconds) and `X-SmartSquad-Signature: sha256=<hex>`. The signature is HMAC-SHA256 with your shared secret over `<timestamp>.<request body>`. Verify it, and reject requests whose timestamp is too old.

How Smart Squad treats your endpoint's response:

RESPONSE	SMART SQUAD DOES
2xx	Marks the batch delivered and removes it from the queue.
408 , 429 , 5xx , timeout, connection failure	Retries with increasing intervals. A <code>Retry-After</code> header is honoured.
413 (payload too large)	Splits the batch and retries.
401 , 403 or another 4xx	Pauses delivery for your agency and alerts Faction Four. Events stay queued.
3xx	Treated as a configuration error. Redirects are never followed.

Collectors known to fit this model include the Splunk HTTP Event Collector (`splunk-hec`), Sumo Logic HTTP sources and Logstash's `http` input with the `json_lines` codec (`ndjson`), and any other HTTPS input that accepts newline-delimited JSON. Microsoft Sentinel's Logs Ingestion API is not supported directly in this release; use an intermediate collector.

Onboarding

1. Your agency provides the endpoint URL, credentials and body format, and allows the Smart Squad hosting address(es) through its firewall.
2. Faction Four configures delivery for your agency.
3. Faction Four sends a **test event**, `siem.test` (`event.id` 5090). Search your SIEM for the `event.uid` Faction Four gives you to confirm the path end to end.
4. Delivery goes live; your agency's events start flowing within about a minute.

SIEM PARSING

Treat each event as JSON: the Windows Event Log `Message` field on-premise, or each line of the request body with HTTPS push. We recommend promoting the following fields to indexed/searchable:

- `event.id` , `event.uid` , `event.name` , `event.category` , `event.outcome`
- `actor.user_id` , `actor.username` , `actor.agency_id` , `actor.ip`
- `subject.type` , `subject.id`
- `correlation_id` (when present)

These give you a strong baseline for detection rules without custom schemas.

VOLUME AND RETENTION

Rough order-of-magnitude expectations for one typical agency:

- 200 active users produce approximately 50–100 MB of raw event data per day
- On-premise, a 512 MB channel with overwrite-as-needed gives roughly 5–10 days of local buffer before SIEM forwarding, which is comfortable for typical collection intervals
- With HTTPS push, events wait in the Smart Squad queue while your endpoint is unavailable, and are set aside for re-sending after 72 hours

Your SIEM is the system of record for long-term retention, in both models. Smart Squad does not keep a copy for you: on-premise, the Event Log channel keeps only what fits under the size cap you set, and with HTTPS push, queued events are deleted once your endpoint has accepted them.

PRIVACY AND COMPLIANCE POSTURE

The SIEM feed is designed with CJIS-style requirements in mind:

- **No credentials cross the boundary.** Passwords, OTPs, tokens and certificate material are never in the SIEM payload.
- **No record content crosses the boundary.** Notes, reports, AI outputs, attachments and form field values stay in the Smart Squad database.
- **Identifiers, not values.** Person and vehicle queries log that a query occurred and, for single-target lookups, which record ID was accessed. They never log the search terms.
- **Sensitive configuration is redacted.** Configuration changes log the key name and, for non-sensitive keys, the old and new values; keys matching sensitive patterns have their values redacted.
- **Failures carry reason codes.** Failed events include a structured reason (e.g. `bad_password` , `invalid_otp`) so detection rules can apply thresholds without parsing free text.
- **HTTPS push is encrypted and isolated** (2.9.5). Events travel over TLS 1.2 with full certificate validation. Each agency receives only its own events. Endpoint URLs and credentials are stored encrypted. Credentials never appear in logs, and operational logs show only the endpoint's host name.

WHAT YOU GET ON DAY ONE

With either delivery model in place, your SIEM gains visibility into:

- Successful and failed sign-ins, with IP, user agent and the reason for any failure (the event catalog lists which failures are reported)
- Lockout events when an account exceeds the failed-login threshold
- Every MFA challenge issued, and every MFA success and failure
- Every password reset request and completion
- Note, report, ticket and intel posting reads, and the changes to notes, reports, tickets, intel postings and form submissions listed in the event catalog
- Every person and vehicle query against the RMS
- Every face search (with confidence level and result count, never the search image)
- Every AI-assisted action on a report (the SIEM sees that an AI activity occurred and against which report; model usage details stay in the Smart Squad database)
- Every user account created, enabled or disabled, every role and capability change, and every system configuration change made in the admin portal (the event catalog lists exactly which administrative changes are reported)

If you build correlation rules, such as "more than N failed logins in M minutes from one IP" or "a face search followed by no record access", those rules have a stable, structured feed to operate against.

QUESTIONS

For implementation support, to arrange HTTPS push onboarding, or to discuss extending coverage (authorization events, file downloads, integration calls), contact your Smart Squad support representative.