CrowdStrike Parsing Standard

Last updated: May 30, 2025

Overview

Easily ingest, parse, and normalize all third-party data with the CrowdStrike Parsing Standard (CPS). CPS helps you better analyze, visualize, and correlate the data represented in events that are detected in your environment.

Requirements

Requires one or more of these subscriptions:

- Falcon Next-Gen SIEM or Falcon Next-Gen SIEM 10GB
- Falcon Complete powered by Next-Gen SIEM

Default roles:

- Falcon Administrator
- NG SIEM Administrator
- NG SIEM Analyst
- NG SIEM Analyst - Read Only
- NG SIEM Security Lead

CrowdStrike clouds: Available in US-1, US-2, EU-1, and US-GOV-1

Understanding the CrowdStrike Parsing Standard

What is CPS?

Streamline data analysis with the CrowdStrike Parsing Standard (CPS) for normalized and standardized event data from third-party sources.

The Crowdstrike Parsing Standard builds on the Elastic Common Schema (ECS), a mature and proven common schema for metrics, logs, traces, and resources. For more info about ECS, see Elastic Common Schema (ECS) 8.x [https://www.elastic.co/guide/en/ecs/8.13/index.html].

Upon ingestion, data is parsed and can be mapped into using CPS, which provides a common language for correlation and analysis of data from different sources. The CPS-parsed data might include vendor-specific alerts, events, and indicators. Vendor-specific telemetry is also preserved and stored because it can contain relevant information for investigation and response.

CPS is designed to normalize and standardize event data for improved analysis, visualization, and correlation. By adopting a common schema, CPS enables you to simplify your search experience, alleviate data complexity, and gain deeper insights into security events found in your environment.

Parsing options

For many data sources, Next-Gen SIEM provides a library of third-party connectors where the parsing and mapping is done automatically.

For any other data sources, you must provide parsers and mappings. For more info, see Ingest Third-Party Data [/documentation/category/fc47f9d5/next-gen-siem/third-party-integration-and-data-connectors].

Normalizing data with CPS

Extracting fields and tags during the parsing stage is essential for search performance.

Mapping different vendor field names, such as timestamp or IP, during the parsing stage to a unified field name helps you simplify your queries. Instead of switching between different fields that contain common information, such as `ip_source` and `source-ip` depending on the log format, you can use consistent field names across different vendors and log formats.

CPS-compliant parsers

CPS-compliant parsers use scripts written in CrowdStrike Query Language (CQL) to transform incoming data into searchable events that trigger detections in Next-Gen SIEM.

You can use default parsers to parse incoming data from common formats provided by vendors. To parse incoming data from other formats, create and manage your own custom parsers. For more info about creating and managing custom parsers, see Parsers [/documentation/page/n00d51ed/parsers].

CPS differs from ECS in a number of ways that build on the specifics of the LogScale core architecture. For example, parsers following CPS make all fields in a log event available as actual LogScale fields, even if they don't match a field in ECS. For more info about the differences, see Differences between CPS and ECS parser fields [/documentation/page/u05f69c9/crowdstrike-parsing-standard#y6c059b2].

For any given data source, when developing a parser, you can decide which domain-specific fields are applicable to the data.

To use CPS-compliant parsers:

- Use CPS-compliant parser tags:
    - `#Cps.version`
    - `#ecs.version`
    - `#event.dataset`
    - `#event.kind`
    - `#event.module`
    - `#event.outcome`
    - `#observer.type`
    - `#Vendor`
- Your parsers must populate the CPS-compliant parser fields. For more info, see CPS compliant parser fields [/documentation/page/u05f69c9/crowdstrike-parsing-standard#x9e84f7b].
- Manage any fields that don't match ECS fields. For more info, see Managing non-ECS fields [/documentation/page/u05f69c9/crowdstrike-parsing-standard#hc94ee02].
- Add any additional fields you need beyond the standard fields. For more info, see Adding fields to CPS [/documentation/page/u05f69c9/crowdstrike-parsing-standard#q96e19e6].

CPS compliant parser fields

This table defines the CPS-compliant fields that are created during parsing

Fields	Descriptions
Event categorization fields	`event.outcome` is only assigned when an event can logically contain an outcome. `event.type` and `event.category` are assigned as LogScale arrays. The arrays can be empty. For more info about event categorization fields, see ECS Categorization Fields [https://www.elastic.co/guide/en/ecs/current/ecs-category-field-values-reference.html].
`#Cps.version`	Contains a `MAJOR.MINOR.PATCH` version number following semantic versioning. This version denotes the version of the standard that the parser targeted during ingestion.
`#ecs.version`	Contains the version of ECS that is being followed by the parser.
`#event.dataset`	Contains the specific name of the dataset within the module described by `#event.module`, prefixed by the value of `#event.module`with a dot in between. This field is only created if it contains information that is not already present in `#event.module`. An example combination of the possible fields in a parser: • #Vendor = microsoft • #event.module = azure • #event.dataset = azure.entraid **Note:** After ingest, all the fields in LogScale are prefixed with #.
`#event.kind`	Contains the kind of the event. This field defines the highest categorization field in the hierarchy.
`#event.module`	The name of the product or service that the event belongs to. Reuse existing values when possible.
`#observer.type`	This field defines the type of observer that the data is originating from.
`Parser.version`	Contains a `MAJOR.MINOR.PATCH` version number following semantic versioning. This version number is specific to the parser that processed the event and is not related to the version of the package from which the parser was installed. The version number is updated according to these rules: • Any change to an existing field, whether large or small, is a breaking change. This requires a new major version. • If new fields are added, then a new minor version is usually enough to reflect the change. • Patch versions are for parser changes that do not affect which fields are output by it, such as performance optimizations, bug fixes, and so on.
`#Vendor`	If the event was parsed with a parser from a package, the vendor name used here must match the vendor name used in the package scope. For example, `fortinet` is used for `fortinet/fortigate`. If a parser sets any of the following fields, those must be consistent with vendor names used in other CPS-compliant parsers: • observer.vendor • vulnerability.scanner.vendor • device.manufacturer

Differences between CPS and ECS parser fields

CPS-compliant parser fields deviate from ECS in these ways:

- Fields that parsers use as tags have their names prefixed with # during ingestion.
- The field `event.original` is not present because LogScale uses `@rawstring` instead.
- The field `event.ingested` is not present because LogScale uses `@ingesttimestamp` instead.
- The field`@timestamp` contains a Unix timestamp, rather than a human-readable timestamp.
- The field `event.code` is not present. The value from `event.code` can still be available to use in a vendor-specific field, such as `Vendor.event_type`.
- Related fields are not included. For more info about these fields, see Related Fields [https://www.elastic.co/guide/en/ecs/current/ecs-related.html].
- The following fields have lowercase values:
  - `*.address`
  - `*.domain`
  - `email.*.address`
  - `host.hostname`
  - `*.hash.*`

For more info about ECS fields, see ECS Categorization fields [https://www.elastic.co/guide/en/ecs/current/ecs-category-field-values-reference.html]. For more info about metadata fields, see Event Fields [https://library.humio.com/data-analysis/searching-data-event-fields.html#searching-data-event-fields-metadata].

Managing non-ECS fields

CPS-compliant parsers make all log event fields available as LogScale fields, even if they don't match an ECS field.

- If there are fields from events that don't exist in ECS, their names are prefixed with `Vendor.`
  - This approach gives ECS fields the `root namespace`, while vendor-specific fields are consistently prefixed with `Vendor`.
- If a field can exist as both an ECS field and a vendor-specific field, the following logic applies:
  - When an ECS field and a vendor-specific field have identical values, the ECS field is preserved and the vendor-specific field is discarded.
  - If the value of the fields differ, both fields are preserved. For example, an ECS field might require its value be lowercase, but the original log has mixed casing. In that event, the vendor-specific field contains the original, mixed-case value.

Adding fields to CPS

When creating or editing a custom parser, you can add new CPS-compliant fields.

- All fields not directly from ECS must have a capital letter starting at the point where they differ from the schema. For more info about creating and managing custom parsers, see Manage parsers [/documentation/page/a76b8289/data-connectors#v43a2825].
  - Using capital letters for field names follows ECS guidance about how to add event fields outside the schema. For more info, see Custom Fields in ECS [https://www.elastic.co/guide/en/ecs/current/ecs-custom-fields-in-ecs.html].
    - For example, `Parser.version` is a fully custom field that is similar to `#ecs.version`, but it includes the CPS-specific prefix `Parser`, which requires a capital letter.
    - An example of extending ECS with a custom field is `observer.Fictional_field`. Here, `observer` is an existing namespace in the schema, but `Fictional_field` is a custom CPS-specific field within that namespace.

Parser guidelines

Apply the following guidelines when creating a parser:

- Define test cases that exercise all parser logic.
  - Personally identifiable information (PII), such as IP addresses, URLs, user names, and email addresses, are not allowed for use in packages.
  - Use valid test data instead of PII. For more info, see Parser sample data [/documentation/page/u05f69c9/crowdstrike-parsing-standard#lc3010e5].
  - Avoid including third-party assets, such as company names and domain names, unless they are directly related to the test.
- Add comments that fully describe the parser logic. For more info, see Example Parser Logic [https://library.humio.com/logscale-parsing-standard/pasta-parser-guidelines-parser-yaml.html].
- Use CPS-compliant fields to set individual parser fields. For more info, see CPS compliant parser fields [/documentation/page/u05f69c9/crowdstrike-parsing-standard#x9e84f7b].
- Create custom parsers based on the following assumptions:
  - They receive one event at a time.
  - Ensure events are not in bulk when they reach the parser.
  - When logs are sent in bulk to the parser, the parser needs to be customized to split them into separate events.
  - They receive events that are not wrapped in custom transport layers. For example, if logs are wrapped in a layer of JSON, the parser needs to be customized to remove the data wrapper.
- Use the parser template when creating parsers. For more info, see Parser template [/documentation/page/u05f69c9/crowdstrike-parsing-standard#n74dcdb1].
- Ensure that any explicitly supported transport methods align with these guidelines.

Ensure that any explicitly supported transport methods align with these guidelines.

Parser sample data

We strongly recommend including sample data for tests in parsers and example data in your packages for demonstration purposes. Review the following info for guidelines on how to create good sample data for your packages. For more info, see Wikipedia's Placeholder names [https://en.wikipedia.org/wiki/Placeholder_name].

Human names

We encourage using John Doe and Jane Doe in sample data.

Company names

Acme is a common company that you can use in sample data, you can also use Octan.

Domain names

This table presents a selection of example domain names to demonstrate typical formats and structures.

Name	Purpose
`example.com`	Completely generic domain that's reserved for example data. For more info, see RFC 2606 [https://www.iana.org/go/rfc2606].
`example.net`	Completely generic domain that's reserved for example data. For more info, see RFC 2606 [https://www.iana.org/go/rfc2606].
`example.org`	Completely generic domain that's reserved for example data. For more info, see RFC 2606 [https://www.iana.org/go/rfc2606].
`*.example`	Top level domain reserved for somewhat relatable data, such as acme.example.com
`mitre.org`	Since Mitre is referenced in many security use-cases, it is okay to use in sample data.

E-mail addresses

E-mail addresses can include almost any combination of human names and domain names, as long as they align with general rules for email addresses. See this table for some common examples.

Address	Purpose
`johndoe@example.com`	A random, private person.
`janedoe@acme.example`	An employee of a company named Acme.

Generic addresses

Common generic local parts are also allowed. This list is not exhaustive, but it is meant to provide guidance.

Address	Purpose
`noreply@example.com`	Commonly used email company mailbox
`info@example.com`	Commonly used company public mailbox
`mailer-demon@example.com`	System mailbox
`postmaster@example.com`	System mailbox
`abuse@example.com`	Abuse mailbox
`webmaster@example.com`	Homepage author mailbox

IP addresses

While IP addresses are public information, we recommend not to use any public routable IPs. The unknown history of IP addresses and lack of control over associated data usage pose potential risks. To mitigate these risks, we advise using non-routable IP addresses, ensuring they cannot be externally accessed.

Private networks

For local networks, we recommend using RFC 1918 networks.

Block	Purpose
`10.0.0.0/8`	Private-Use Network
`172.16.0.0/12`	Private-Use Network
`192.168.0.0/16`	Private-Use Network

169.254.0.0/16, fe80::/10	Link-Local addresses
224.0.0.0/4	Multicast network. Typically used for devices to locate eachother behind the same firewall
fc00::/7	Unique Local Addresses
ff00::/8	Multicast Addresses RFC 4291 [https://www.rfc-editor.org/rfc/rfc4291]

Public networks

For public networks, we generally only allow TEST-NET blocks, with a few exceptions. The list of exceptions is growing, so direct any requests for additional networks to humio_packages@crowdstrike.com.

> **Note:** We generally only allow IPs of services that are specific to an IP address, such as DNS.

Block	Purpose
192.0.2.0/24	TEST-NET-1 For more info, see RFC 5737 [https://www.rfc-editor.org/rfc/rfc5737]
198.51.100.0/24	TEST-NET-2 For more info, see RFC 5737 [https://www.rfc-editor.org/rfc/rfc5737]
203.0.113.0/24	TEST-NET-3 For more info, see RFC 5737 [https://www.rfc-editor.org/rfc/rfc5737]
1.1.1.1, 1.0.0.1	CloudFlare DNS
8.8.8.8, 8.8.4.4	Google DNS
2001:db8::/32	Reserved for documentation and examples
2001:4860:4860::8888, 2001:4860:4860::8844	Google DNS
2606:4700:4700::1111, 2606:4700:4700::1001	CloudFoundry DNS

Parser template

Use this template to create a correctly defined parser.

```
`
// #region PREPARSE
/************************************************************
****** Parse timestamp and log headers
****** Extract message field for parsing
****** Parse structured data
************************************************************/
// #endregion
// #region METADATA
/************************************************************
****** Static Metadata Definitions
************************************************************/
| ecs.version := "8.11.0"
| Cps.version := "1.0.0"
| Parser.version := "1.0.0"
| Vendor := ""
| event.module := ""
| event.dataset := ""
// #endregion
// #region NORMALIZATION
/************************************************************
****** Parse unstructured data (i.e. message field)
****** Normalize fields to data model
************************************************************/
// #endregion
// #region POST-NORMALIZATION
/************************************************************
****** Post Normalization
****** Custom parser logic needed after normalization
************************************************************/
// #endregion`
```

Vendor guidelines

Apply the following guidelines when creating a package:

- For existing vendors:
  - Before creating a new vendor name, check to see if your vendor name already exists. For a list of vendor names, see

Vendor Guidelines [https://library.humio.com/logscale-parsing-standard/pasta-vendors.html].

- If the vendor already exists in other packages, use the previously defined name.
- Maintaining consistency across packages is crucial for proper categorization and searchability.

- When adding a new vendor, follow these principles:
  - Choose concise and clear names that accurately represent the vendor.
  - Use full words or official acronyms instead of abbreviations or legal entity names. For example, use "apple" instead of "Apple Inc." or "aws" instead of "amazon".
  - Choose names that will remain relevant and recognizable over time.
- Vendor name requirements:
  - Must contain at least three characters.
  - Use only lowercase letters (a-z), numbers (0-9), underscores, and hyphens.

Note: These requirements ensure compatibility with our systems and maintain a consistent naming convention. The vendor name you choose also serves as the scope for your package. This means the vendor name is a key identifier in the package structure.

Review this table for examples of #Vendor tags that follow CPS vendor guidelines.

#Vendor	Legal name
akamai	Akamai Technologies, Inc.
apple	Apple Inc.
cisco	Cisco Systems, Inc.
dell	Dell, Inc.
zscaler	Zscaler, Inc.

Module guidelines

Apply the following guidelines for module names when creating a package:

- If a module name is already used in other packages, reuse the same name for consistency.
- If you are adding a new module name make sure that it is clear and concise.

Review this table for examples of module names that are used in parsers for the #event.module tag.

#Vendor	#event.module
abnormal	emailsecurity
akamai	asec
broadcom	proxysg
dell	isilon
zscaler	zia

Deprecated parsers

CrowdStrike maintains a number of default parsers. When a parser is marked deprecated, we no longer maintain and update that parser. We strongly recommend using the latest version of default parsers for your data connector. Using a deprecated parser won't stop data ingestion but can result in deteriorated detection coverage.

To ensure best detection coverage for your environment, follow these steps:

1. Ensure that all correlation rules, dashboards, scheduled searches, and saved queries running in your environment that use the #type=<parser_name> field are updated to use the CPS compliant parser fields [/documentation/page/u05f69c9/crowdstrike-parsing-standard#x9e84f7b] #Vendor=<vendor_name> | #event.module=<product_name>. For example, if you have a saved query that uses the #type=vectra-ecs, update this query to use the fields #Vendor=vectra | #event.module=brain instead. If a vendor supports more than one product, you can run queries for multiple products that use the same parser. For example, to get results for multiple Akamai products that are using the akamai-zerotrust parser, run this query: #Vendor=akamai | #event.module=eaa OR #event.module=sia OR #event.module=mfa OR #event.module=guardicore.

   Tip: To look up #Vendor and #event.module fields for a parser, go to **Next-Gen SIEM > Log management > Advanced event search [/investigate/search]** and search groupBy(#Vendor). Use the vendor value returned and search for #Vendor=<vendor_value> | groupby(#event.module). For example, to get the #Vendor CPS field for the Vectra AI parser, search for groupBy(vectra) which returns the value vectra and to get the #event.module field values, search #Vendor=vectra | groupBy(#event.module).

2. Ensure that your data connectors are using the latest parsers. To update your connector, see Edit a connection [/documentation/page/a76b8289/data-connectors#f729862b]. To see the default parser for each connector, see Third-Party Data Sources [/documentation/category/je6a45b3/next-gen-siem/third-party-integration-and-data-connectors/third-party-integrations].

This table lists deprecated parsers and the corresponding default parsers we recommend for your data connectors:

Deprecated parser	**Default parser** (Recommended)
1password	1password-enterprise
abnormal_security_ecs	abnormal-emailsecurity
alteon-syslog	radware-alteon
apm-syslog	f5networks-bigip
asec-json	akamai-asec
asimily-iomt-json	asimily-iomt
azuread-ecs	microsoft-azure-ad
cef-latest	claroty-ctd
centrix-iot-json	armis-centrixiot
cisco-ise-syslog	cisco-ise
cisco_seg_ecs	cisco-seg
ciscoasa-ecs	cisco-asa
ciscoumbrella	cisco-umbrella
citrix-netscaler-syslog citrix-netscaler-waf-cef	citrix-netscaler-adc
clearpass-syslog	aruba-clearpass
cloudflareone-ecs	cloudflare-one
cloudtrail	aws-cloudtrail
corelight-ecs	corelight-ids
corelight-json	corelight-ids
cwaf-cef	imperva-cloudwaf
deception	zscaler-deception
dlp-cef	forcepoint-dlp
duo-activity-json duo-admin-json duo-authentication-json duo-telephony-json duo-trustmonitor-json	cisco-duo
extrahop-ecs	extrahop-revealx360
fireeye-nx	trellix-fireeye-nx
firepower-syslog	cisco-firepower
forgerock-ecs	forgerock-identity
fortimail	fortinet-fortimail
fortinet-ecs	fortinet-fortigate
fsx-xml	aws-fsx
Google_Chrome_Enterprise	google-chrome-enterprise
guardduty-json	aws-guardduty
haproxy-syslog	haproxy
isilon-syslog	dell-isilon
island	island-enterprisebrowser

menlo-ecs	menlo-msip
microsoft_defender	microsoft-defendero365-graphapi
mimecast-ecs	mimecast-emailsecurity
ms-defender-graph-ecs	microsoft-defendero365-graphapi
ms-defender-stream-ecs	microsoft-defendero365-eventhubs
microsoft-windows-dhcpserver	microsoft-windows-dhcp-server
netskope-ecs	netskope-sse
nozomi-syslog	nozomi-ids
obsidian-json	obsidian-securitydata
okta-ecs	okta-sso
onelogin-json	oneidentity-onelogin
paloalto-ecs	paloalto-ngfw
paloalto-firewall-syslog	paloalto-ngfw
pfsense-syslog	netgate-pfsense
ping-ecs	pingidentity-pingone
prisma-sd-wan	paloalto-prisma-sdwan
proofpoint-tap-ecs	proofpoint-tap
rubrik-json	rubrik-securitycloud
s3access-space-delimited	aws-s3serveraccess
skyhigh-ecs	skyhigh-sse
srx-syslog	juniper-srx
syslog-utc	broadcom-proxysg
syslog-utc	cisco-ios
syslog-utc	infoblox-nios
sysmon	microsoft-sysmon
tausight-json	tausight-ephi
umbrella	cisco-umbrella
vectra-ecs	vectra-brain
vmware-esxi-ecs	vmware-esxi
vpcflow_default	aws-vpcflow
waf-json	aws-waf
windows-dhcp-client	microsoft-windows-dhcp-client
windows-dhcp-server-csv	microsoft-windows-dhcp-server
windows-dns	microsoft-windows-dns
zerotrust-json	cloudflare-one
zscalernss-dns zscalernss-fw zscalernss-tunnel zscalernss-web	zscaler-internetaccess
zscaler-ecs	zscaler-internetaccess
zscaler-zpa-app-connector-status-json zscaler-zpa-app-protection-json zscaler-zpa-audit-json	

zscaler-zpa-audit-json zscaler-zpa-browser-access-json zscaler-zpa-user-activity-json zscaler-zpa-user-status-json	zscaler-privateaccess
ai_analyst_alert-syslog model_breach_alert-syslog system_status_alert-syslog	darktrace-detect