exabeam

Exabeam How Content Works Guide

Exabeam Data Lake and Advanced Analytics

Publication date April 3, 2021

Exabeam
2 Waters Park Dr. Suite 200
San Mateo, CA 94403

650.209.8599

Have feedback on this guide? We'd love to hear from you!
Email us at docs@exabeam.com

Disclaimer: Please ensure you are viewing the most
up-to-date version of this guide
by visiting the Exabeam Community.

Copyright

All content in this document, including text, graphics, logos, icons, images, and video clips, is the exclusive property of Exabeam or its content suppliers and is protected by U.S. and international copyright laws. The compilation (meaning the collection, arrangement, and assembly) of all content in this document is the exclusive property of Exabeam and is also protected by U.S. and international copyright laws. The content in this document may be used as a resource. Any other use, including the reproduction, modification, distribution, transmission, republication, display, or performance, of the content in this document is strictly prohibited.

Copyright ©2021 Exabeam, Inc. All Rights Reserved.

Trademarks

Exabeam, the Exabeam logo, Threat Hunter, Smarter SIEM, Smart Timelines and Security Management Platform are service marks, trademarks or registered marks of Exabeam, Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. The marks and logos displayed in this document may not be used without the prior written consent of Exabeam or their respective owners.

Patents

Exabeam owns, and reserves all rights for, patents for Exabeam products and services, which may be protected under registered patents as well as patents pending.

Other Policies

For information regarding Exabeam’s treatment of personally identifiable information, please review Exabeam’s current privacy policy at www.exabeam.com/privacy.

Table of Contents

1. Introduction to How Content Works

These topics will help you understand the following Exabeam content areas:

- **Understanding the Log**
  - When to parse ingested logs in Data Lake and Advanced Analytics
  - Information that is provided in a log and the minimal critical fields
  - Mapping a log to an Exabeam event
  - Example log process
- **Parsers**
  - Associating a log with a parser
  - Extracting values using regular expressions
  - Parser parameter definition
  - Testing a parser on Advanced Analytics
- **Event Building**
  - Matching parsers to event builders
  - Event builder configurations
  - Event stitching
- **Enrichment**
  - Types of enrichment
  - Enrichment use cases
  - Event enricher configurations
- **Persistence and Templates**
  - Persisting a field
  - Adding a field in an event template
  - Custom persistence and templates
- **Models**
  - Types of models
  - Model attributes
  - Model categories
- **Rules**
  - Types of rules

- Creating rules
- Rule attributes

2. Understanding the Log

Exabeam can ingest logs directly from the source, fetch logs from SIEM log repositories, or ingest logs via Syslog as well as Exabeam Data Lake. Logs provide insight into the activity of users and entities (such as servers and workstations) and security issues across your enterprise. Context sources give our platforms additional information outside of the log data to make sense of the logs.

Once logs have been collected, they can be parsed in Data Lake and Advanced Analytics. This section will help you determine when to parse the logs and how to identify fields within the logs.

2.1. Do You Need a Parser?

In order to determine whether you need a parser, first determine whether your log qualifies for security processing and analysis. This is especially pertinent when parsing for Data Lake. For Advanced Analytics, all ingested logs must be parsed.

A parser extracts values from a log and maps those values to the appropriate Exabeam fields. See Exabeam Parsers for more information.

2.1.1. PARSING FOR DATA LAKE

Data Lake is designed to work with any log source and does not require a parser for much of its functionality. Data Lake can ingest, index, and search all logs even if they are not parsed.

Since parsing is a complex and resource intensive piece of the Data Lake pipeline, it may have some performance implications . Therefore, you should only use parsers where necessary . It is important to note that you can always go back in Data Lake and reparse old logs.

In Data Lake, you can do the following without a parser:

- Send logs to Data Lake (ingest and index)
- Set log retention
- Perform string-based searches
- Create rules on your data
- Create certain reports and dashboards

> **NOTE**
> You are limited in the types of rules, reports, and dashboards you can create without a parser since you do not get the benefit of field values.

If the data is parsed, these additional features are available:

- Field specific reports and dashboards
- Field specific visualizations
- Field specific rules
- Add context to logs, which make them searchable

2.1.2. PARSING FOR ADVANCED ANALYTICS

Conversely, in Advanced Analytics, only parsed logs can be ingested. However, only logs related to an event type that Advanced Analytics can process and analyzes should be parsed. For a list of possible Advanced Analytics events and the content in which they are used, please refer to the Advanced Analytics Content Guide.

In logs that support Advanced Analytics event types, only the specific fields that are used for display or for processing need to beparsed. For example, for Entity Analytics logs like network connection, events do not need a user, and only IP, host, port, and similar information should be parsed.

If logs are required for Advanced Analytics (UBA) , they must have a user or a way to identify the user, such as a badge to a user list. Without a user, Advanced Analytics cannot process the log and it does not make sense to create a parser.

NOTE
In Entity Analytics, events without a user can be processed, such as network connection status and netflow connection. Any event type can go on an asset timeline as long as it has a hostname. It is still necessary to make sure the log falls into one of the Advanced Analytics security related event types.

2.1.2.1. Minimum Log Data for Processing in Advanced Analytics

Once you have determined the log is security related, the next question to ask yourself is, does the log event have the minimum required information to be processed in Advanced Analytics.

For an event to be processed in Exabeam, it needs either:

- A username (typically AD account), or information from which it can be derived, such as an email or distinguished name badge ID
- Information about the originating or receiving asset (hostname or IP)

In addition to the minimum information, it also helps for the log to have a full timestamp and information about the host (hostname or IP) which produced the log. In Exabeam, these values will be parsed as "time" and "host", respectfully. The time information must include the year, month, day, hour, minutes, seconds, and preferably a timezone of when the log was generated.

2.1.2.2. Identify the Exabeam Event

Once an event matches the minimum criteria the next question would be which Exabeam event type best describes the event. An event type can be for example "remote-logon", "app-activity”, etc. For a list of Advanced Analytics event types, please refer to the Advanced Analytics Content Guide.

NOTE
The log may not match an Exabeam event type exactly. In that case, you should use the Exabeam event that represents the log most closely.

2.2. Log Process Example

The following is an example of decision process for choosing logs for Advanced Analytics. This includes general questions to ask, and answers based on this example data:

```
"Aug 14 22:13:03 10.130.168.57 vendor=Forcepoint product=Security
product_version=8.3.0 action=permitted severity=1 category=1913
user=jdoe@company.com src_host=10.130.164.49 src_port=49265 dst_host=host.com
dst_ip=2.2.2.2 dst_port=443 bytes_out=0 bytes_in=4805 http_response=0
http_method=CONNECT http_content_type=- http_user_agent=Mozilla/
5.0_(Windows_NT_6.1;_WOW64)_AppleWebKit/537.36_(KHTML,_like_Gecko)_Chrome/
59.0.3071.109_Safari/537.36 http_proxy_status_code=200 reason=%<reasonString>
disposition=1028 policy=Exceptions_and_Filter_Updates**BasicBlocking role=1807
duration=4 url=https://exampledomain.com/download/virus.exe"
```

1. Determine whether the log has security significance. In this case, the log is from a web proxy product which shows access to websites, and therefore has security significance.
2. Answer the following questions to perform an initial analysis of the log:
   a. Can the log be tied back to a user or device?
   Yes, the log can be tied back to a user (`user=jdoe@company.com`).
   b. Does the log have a complete time field?
   No, the log time is missing the year. We will use exabeam_time (`Aug 14 22:13:03`).
   c. What product or vendor produced these logs?
   (`vendor=Forcepoint`) It would also help to know the product, but it is not crucial in this case.
3. Perform the secondary analysis, asking whether the log can be mapped to an Exabeam Event Type.
   a. The event type for this log is "web-activity-allowed". We expect "web-activity-allowed" logs to have "web_domain" and user information, which this log includes.
4. Based on these questions, we can conclude that Exabeam will provide value by ingesting this log.

3. Exabeam Parsers

A parser is a configuration in the `parser.conf` file that defines:

- The logs to extract values from
- Which values should be extracted from the log
- The Exabeam fields these values are mapped to

NOTE
The Exabeam Advanced Analytics pipeline is as follows:

parsing > event building > enrichment > session building > modeling > rule triggering

Once you have determined that a log event is valuable for security analytics, the next phase is to extract the values of interest from the log and map them to Exabeam fields.

This is done by the parsing stage, which is the very first stage of the analytics engine pipeline. From ingesting logs to scoring on a timeline, parsing is the entryway into Advanced Analytics.

3.1. Associate a Log with a Parser

The parsing engine associates a log with the correct parser by using a unique string or strings that exist only in the specific log. These strings are specified in the `Condition` parameter of the parser. If multiple conditions are specified, all of the conditions must exist in the log for the parser to take effect.

Parser conditions are evaluated according to their order in the parser list. A log entering the ingestion engine will first be checked against the conditions of the parser at the top of the file. If none match, then it moves onto the next parser in the file, and so on.

NOTE
Once a log is caught by a parser, no other parser conditions will be evaluated. The parser with the matched condition will be used to parse the event.

If parsers have similar conditions, you must place the parser with a broader condition below the parser with a more specific condition. Otherwise, the parser with the broader condition will also parse the more specific logs.

3.2. Extracting and Mapping Values

Regular expressions, or *regexes*, allow Exabeam to extract specific patterns from logs and map these values to fields based on the order the regexes are applied. A regex for a value of interest will be surrounded by parentheses. The first value in the parentheses will be a set of curly brackets containing the name of the field of the extracted value. The curly brackets are followed by the regular expression identifying the value.

For example, the expression ABC`({my_field}...)` will parse the immediate three characters after the string “ABC” in the log and will map them to a field called `my_field`. For example, if the received log is "ABC123XYZ" the field `my_field` will contain the value “123”.

If the string "ABC" does not exist in the log, the field `my_field` will not be created.

All regular expression statements are evaluated in consecutive order against the entire log. If a value is mapped to a certain field in one expression and then a different value is mapped to the same field, the second mapping will overwrite the first.

3.3. Parser Parameter Definition

The following is an example parser parameter definition that contains common fields, such as Name, Vendor, and Product.

```
{
  Name = o365-inbox-rules-2
  Vendor = Microsoft
  Product = Office 365
  Lms = Direct
  DataType = "app-activity"
  TimeFormat = "yyyy-MM-dd'T'HH:mm:ss"
  Conditions = ["""Operation":"Set-Mailbox""" ]
  Fields = [
    """"CreationTime":"({time}\d\d\d\d-\d\d-\d\dT\d\d:\d\d:\d\d)"""",
    """Forward.+?Value":"(smtp:)?({target}[^"]+@({target_domain}[^"]+))""""
    """"ResultStatus":"({outcome}[^"]+)"""",
    """"ClientIP":"\[?({src_ip}[^"]+?)\]?:({src_port}\d+)"""",
    """({activity}Set-Mailbox)""",
    """cs1=(\[\{"additional-properties"\:)?\{"({activity}[^"]+)""",
    """msg=({additional_info}.+?)\s\w+=""",
    """"Value":"(?:smtp:)?.+?@({target_domain}[^"]+)"""",
    """UserId":"({user_email}[^"\\]+@({user_domain}[^"]+))""",
    """destinationServiceName=({app}.+?)\s*filePath"""
    """({app}Office 365)"""
  ]
  DupFields = ["app->resource"]
}
```

3.3.1. PARSER FIELD DESCRIPTIONS

The following table lists and describes parser fields, and whether they apply differently to Data Lake and Advanced Analytics:

Field	Description	In Data Lake	In Advanced Analytics
Name	The name of the parser. You will use this name when creating event builders. You will see this name in `evt.gz` logs as the value for exa-msg-type. Each parser name must be distinct, or a parser with the same name that is seen previously in the configuration files will overwrite any parser that was previously read with the same name.		

Field	Description	In Data Lake	In Advanced Analytics
Vendor	The name of the company or vendor that builds or sells the logging source. In the Parser Parameter Definition example, Office 365 is the log source that generates the activity logs, and Microsoft is the company that builds the product.	The value of this parameter will be in the vendor field, which will be indexed and searchable.	This is searchable from Threat Hunter.
Product	The name of the product that generates these logs.	The value of this parameter will be in the product field, and will be indexed and searchable.	This value is searchable in Threat Hunter.
Lms	This is an optional field used for parser management. It does not have any effect on the parsed log. In the previous example, Direct means the logs are being ingested via syslog directly from the log source, rather than a log management system. Other possible values are DataLake, Splunk, Qradar, and Arcsight, if one of these happens to be the log management systems forwarding logs to Advanced Analytics.	This field has no effect.	This field has no effect.
TimeFormat	A regex-style definition of the structure of the parsed time field. Exabeam supports Unix timestamp formats for parsers, as well as any format that is Unix-readable. If the time field is parsed as a 10-digit number, such as epoch time, then the value for TimeFormat would be `epoch`. In the previous example, we parse time as 2019-10-100T10:12:50.		
Conditions	A set of strings that be included in the logs for the parser to begin evaluating the log. The regexes will be compared against the log only if all conditions are met.		
Fields	All the regexes for this parser, where the fields are actually extracted. For any regex, you can parse as many fields as you want. In the previous example, some regexes parse multiple fields, such as the regex parsing `user_email` and `user_domain`. Fields are parsed in their own regex for performance reasons.		
ISHVF	Ishvf = IsHighVolumeFeed		This field is deprecated as of Advanced Analytics i46. For pre-i46 versions, set this to true ("Ishvf = true") if for the specific logs caught by that parser there is a large volume that is ingested by the ingestion engine and evaluated by that parser for those logs.
DupFields	This is an array that duplicates fields into new field names. It is much more performant than to duplicate the regex. In the previous example, "app" is already parsed by the regexes. You can also create a duplicate field called "resource" with the value of what "app" is parsed as.		

Table 1.

3.4. Test a Parser on Advanced Analytics

To test a parser in Advanced Analytics, run the following command on the Advanced Analytics deployment:

```
(.env)$ exa-fetch-parse --config-file /opt/exabeam/config/custom/
custom_lime_config.conf --request
"(2019-06-03,2019-06-05,syslog)" --status ParseOnly
```

The command will run the ingestion engine over the supplied log fetch type, in this case the logs that were sent by syslog, and will only parse the specified dates. The directory the command runs the ingestion engine on is specified in the configuration file specified by the `--config-file` parameter. This is typically the log storage directory. The output will be the same directory.

3.5. Troubleshooting Regexes

Regexes extract data to ingest into the Exabeam platform. Creating the correct regex is crucial to getting all the value that Advanced Analytics offers, such as rule scoring and modeling.

You can use multiple regexes for a single field name. Typically, this is used when the format of a field differs within a log. In that situation, you can use multiple regexes to be sure that that one of them will parse the field correctly. In the case that both regexes will be matched against the log, the regex that appears later (further below in the fields array) will have higher precedence, and thus its value for the field will be used.

For example, in the Parser Parameter Definition example, two regexes can parse the app field. If the first regex works, and an app value is parsed, and the second regex also works, the data parsed by the second regex will overwrite what was initially parsed for app by the first regex.

> **NOTE**
> You can use regex101.com to help you create and test regex syntax.

3.5.1. REGEXES MISPARSING

Design your regexes to be able to capture all possible variations of your data. By carefully creating and testing your regexes, you can make sure that Advanced Analytics doesn't miss data that would prevent it from being able to model a specific field.

Design your regexes to capture edge cases for how a value might appear in the log. In many cases, regexes are initially built to end when a space or an array/log-constructor-like character (':','[', '}') is used to end the regex. You will need to balance the requirements to allow a broadly tuned regex to capture what should be required, as well as limit how far the regex is allowed to capture. Sometimes a regex change is required due to a space being allowed in a filename, or the log management system happens to be appending forward or back slashes.

3.5.2. PERFORMANCE TUNING

The speed of a regex is crucial for the stability of the ingestion engine. A single high volume log source that hits a single parser that takes 70 ms to parse a single log will severely degrade performance . Starting with Advanced Analytics I48, parsers that impact the ingestion process as a whole will be automatically disabled.

In many cases, this occurs because the regex was designed to be as broadly tuned as possible, and does several 'look aheads' in the log. If a log line is large, a single regex in a parser that tries to look through most, if not the entire log, will cause the ingestion engine to slow down and eventually disable the parser.

3.6. Additional Parser Guidelines

Here are a few very important notes to keep in mind when working with parsers:

- If time is not available in the raw log, use the syslog field headers.
- Without parsing a user, src_ip, dest_ip, dest_host, or src_host , Advanced Analytics cannot process the event and the log will be of no value to you.
- Parsers are organized into major vendors . For example, parsers for logs generated by Carbon Black products can be found in `config/default/parsers_carbonblack.conf`.

4. Exabeam Event Building

The event builder stage is the part of the analytics engine pipeline that categorizes a parsed message into an Exabeam event type. Event types are the basic units that are used by the rest of the processing engine (enrichment, models, rules, and UI components).

NOTE

The Exabeam Advanced Analytics pipeline is as follows:

parsing > **event building** > enrichment > session building > modeling > rule triggering

Every parser is matched to an event builder definition. If there is no event builder for a parser, nothing is done with the parsed output of that log event. Parsed messages that do become events are written into `evt.gz` files.

The event building stage introduces several advantages:

- **Reduces the number of parsers** – Different event types can be created based on a single parser. For example, a Windows login event will indicate the successful or failed outcome of a login in a field, which will have the value 0x0 in case of a successful login and another value in case of a failed login. This value can be parsed and conditioned on the event builder to create a local-logon or a failed-logon event. This eliminates the need for two parsers to capture these two event types.
- **Combines information in two logs** – Some log sources provide all the information needed in an Exabeam event in two separate log events. For example, one VPN log could indicate the user's session and the source IP, and another log could provide the user's session ID, user name, and assigned IP. In order to create a meaningful vpn-login event, the information from both logs have to be combined into a single event. This can be achieved in the event builder based on the session ID field that would be identical in both logs.
- **Complex combination of multiple logs** – Some email sources can generate hundreds of logs for a single email. In order to combine information in all these messages into a single event, a complex logic is needed which can be defined in the event builder.

4.1. Match Parsers to Event Builders

Different event builders can match the output of the same parser to create different event types. This is done to create different event types based on what is seen in the log.

There is also no issue with having a single event builder match the output of several parsers. This is done for example when there are several formats of an event that require several parsers (for example, a Windows event is collected with Snare and Beats from different systems). In fact, this will reduce the number of event builders and make them easier to manage.

However, unlike in the parsing stage where a parser cannot match with an event if the event has already been matched with another parser, multiple event builders can apply to a single message. This results in the creation of multiple events for a single log. Therefore, it is important to make sure that no more than one event builder will match a parsed message. This is usually done by creating a condition on the name of the desired parser as well as some of the data in the message, if necessary. There may be special cases in which more than one event should be created for a single log.

4.2. Event Builder Definition

Here is an example event builder definition containing a number of common fields, such as name, output-type, source, and vendor.

```
netskope-file-write = {
   input-message = [{
     expression = "InList(type, 'netskope-activity','s-netskope-activity','cef-
netskope-file-operation-1') and
InList(toLower(activity),'edit','move','create')"
   }]
   name = netskope-file-write
   output-type = file-write
   source = Netskope Active Platform
   vendor = Netskope Active Platform
}
```

4.2.1. EVENT BUILDER FIELD DESCRIPTIONS

All of the fields below are required.

Field	Description
Event Builder ID	netskope-file-write This can be any value but it must be identical to the name parameter.

Input-message	This contains the expression(s) that should be considered by this event builder. Similar parsers that should create the same type of event based on the same conditions seen across logs can be grouped into a single event builder as shown in the above Event Builder Definition example.
	`"InList(type, 'parser_name_1','parser_name_2')"`
	This expression matches a parser to an event builder. The type field contains the name of the parser that created the message.
	NOTE
	This type refers to the parser message type, which happens to be the parser name.
	Everything parsed by the parsers defined in this expression only get evaluated by this event builder definition (unless the parser also exists in another event builder) and set of conditions.
	`"InList(toLower(activity),'edit','move','create')"`
	The rest of the conditions involve using logical expressions that check against parsed fields to decide whether a message becomes an event.

Name	This is the name of this specific event builder definition. It must be the same as the key given to the entire config/hocon block. If not, the analytics engine will not start. If another event builder contains the same name further down the file, the first one will be overwritten.
Output-Type	This is the Exabeam event type assigned to this event. It has to match one of Exabeam events and will determine how this event will be handled by downstream analytics. **NOTE** This type refers to the event type.
Source	The specific software/OS/product name that generates the log. This field will typically be visible in the UI.
Vendor	The name of the vendor of the system that generated the log.

Table 2.

4.3. Event Stitching

Sometimes multiple logs are needed to build an event, meaning all the data needed for a single event are spread out over multiple logs. Event stitching allows the event builder to extract all the relevant pieces of information from different parsed logs and create a single event. Different parsers extract information from the relevant logs detailing different pieces of the same logical activity, for example a vpn login, to create a single event.

Two types of event builders can be used to combine information from multiple messages to a single event:

- **VariableMessageMultiEventTracker** – Used to combine information from a variable number of messages into a single event.
- **ContivityMultiEventTracker** – Used to combine information from exactly two messages into a single event. Here is an example Postfix email event builder:

```
postfix-email-in = {
  input-message = [ ### for 'ContivityMultiEventTracker' ebuilders there
should only be two objects in this field array, as
'ContivityMultiEventTracker' uses exactly two messages
   {
    expression = "type = 's-postfix-dlp-email'"    ### the expression that
contains what parser to catch and other logical expression as the conditions.
    output-fields =
"msg_id,src_ip,src_host,sender,recipients,recipient,host,subject"  ### fields
to keep from the parsed message that will be apart of the event
    type = s-postfix-dlp-email   ### a name given to this specific message
extraction
   },
   {
      expression = "type = 's-postfix-dlp-email-1'"
      output-fields = "dest_ip,dest_host"
      type = s-postfix-dlp-email-1
   }
  ]
  key-fields = "msg_id" ### the field that ties together the two messages and
should be present in both parsed messages
  name = postfix-email-in ### the name of the event builder, same as the
hocon block name
  output-type = dlp-email-alert-in ### event-type
  source = Postfix  ### product name
  tracker = ContivityMultiEventTracker #### special type of event builder,
other value for tracker is 'VariableMessageMultiEventTracker'
  vendor = Postfix ### vendor name
}
```

4.4. Types of Event Type Fields

When an event builder creates an event from a log, information from that log is mapped to three types of fields: required, extended, and informational.

An event builder always creates an event of a specific type. Each event type has unique fields that correlate to certain information in a log. There are three types of fields: required, extended, and informational.

For an event builder to create an event, a log must contain information that matches an event type's required fields. Information that maps to extended and informational fields is optional to create an event, but is still useful to process and display the event.

4.4.1. REQUIRED EVENT TYPE FIELDS

An event type's required fields ensure that an event has the minimum set of meaningful data for other components to process.

Components, like rules, machine learning algorithms, and Smart Timelines™, need a few basic data points to properly process an event. To ensure that an event contains these data points, Event Builder creates an event from a log only if that log contains the required data for a specific event type.

For example, the `process-created` event type has a `process_name` required field. To create an `process-created` event, a log must contain information about the process name.

4.4.2. EXTENDED EVENT TYPE FIELDS

Information contained in extended event type fields help rules and models detect anomalies.

When Event Builder creates an event from a log, it matches certain information in the log to an event type's extended fields, if the information exist. Risk Engine uses the information contained in the extended fields to train models and evaluate the event against rules.

For example, the `vpn-login` event type has an `os` extended field. The `VPN29 - VPN Operating Systems` model trains on this `os` information. If the model considers the `os` anomalous, it may trigger the `VPN32 - First VPN from OS` rule.

4.4.3. INFORMATIONAL EVENT TYPE FIELDS

Informational event type fields enrich Advanced Analytics events and data in Data Lake with contextual information.

When an event builder creates an event from a log, it matches certain information in the log to an event type's informational fields, if the information exists. In both Advanced Analytics and Data Lake, these fields are used to correlate important data, like host and IP addresses, and enrich events with contextual information so you can easily search for logs, events, users, or assets.

For example, `user_sid` is an informational field for the event type `kerberos-login`. Data Lake maps `user_sid` to `account_id` so you can search for either `user_sid` or `account_id` and find the same log.

In Advanced Analytics, Smart Timeline™ events also display certain information based on these informational fields. If the event builder can't find the information for informational fields in the log, the informational fields appear blank in the Smart Timeline.

5. Exabeam Enrichment

Enrichment refers to the part of the analytics engine pipeline that adds 'contextual' information to an event that is otherwise not already parsed or not available in the log.

> **NOTE**
> The Advanced Analytics pipeline is as follows:
>
> parsing > event building > **enrichment** > session building > modeling > rule triggering

Most enrichment adds new values, modifies them, or creates new fields based on existing fields or context table lookups.

5.1. Types of Enrichment

There are two types of enrichment, system- and user-defined.

5.1.1. SYSTEM-DEFINED

This type of enrichment is done automatically by Advanced Analytics in the backend, and can be slightly tuned by `custom_exabeam_config.conf`.

- **Host-Ip Mapping** – If a user or hostname is detected without the other, this enrichment feature populates the missing field based on previously seen data.
- **Security/Dlp-Alerts-to-User Mapping** – When security or DLP alerts do not have the user information, this enrichment feature populates the user field based on previously seen data.

5.1.2. USER-DEFINED

This type of enrichment can be granularly controlled by the user.

- **Context Enrichment** – Performs a lookup from a context table to populate a field.
- **Event Enrichment** – Modifies/adds/removes fields. This is the most common type of enrichment, defined the same way context enrichment is defined. All logical expressions available in the analytics engine, excluding model/session expressions, can be used in the Event Enricher.
- **Event Duplicator** – Duplicates an event for the purpose of adding to a different user/asset timeline.

5.2. Enrichment Use Cases

It is possible to gather a lot of information about the log from values within the log itself that are not specifically parsed. You can then act on the contextual items seen in the logs, such as in modeling or rule triggering.

- Track Specific Field Values
- Context Retrieval
- New Field Based on Information Within the Event

5.2.1. TRACK SPECIFIC FIELD VALUES

In the following example, we create a new field called 'win_command_count' to allow a rule the ability to "count" within a sequence/session specifically on certain values for process_name.

```
count-win-command {
  EventTypes = ['process-created','privileged-object-access']
  Condition = "exists(process_name) &&
 ((InList(toLower(process_name),'net.exe') &&
InList(toLower(arg),'start','user','time','view','use','localgroup','group','con
fig','share')) || (InList(toLower(process_name),'netsh.exe') &&
InList(toLower(arg),'advfirewall')) ||
(InList(toLower(process_name),'tasklist.exe','ver.exe','ipconfig.exe','systeminf
o.exe','netstat.exe','whoami','qprocess.exe','query.exe','type.exe','at.exe','re
g.exe','wmic.exe','wusa.exe','sc.exe','rundll32.exe','psexesvc.exe',
'icacls.exe', 'arp.exe', 'route.exe')))"
  Map = [
    {
      Field = "win_command_count"
      Value = """'1'"""
    },
    {
      Field = "win_critical_command"
      Value = """process_name"""
    }
  ]
}
```

The analytics engine "count" expressions allow us to count against a certain field, such as "process_name," but the expression does not let us specify for what values of that field we should count on. The below enricher allows us to create a field that only ever exists when it is or contains the values we want to count on. Now we can count how many times the following command was used.

```
Sum(win_command_count, 'process-created')
```

NOTE
When you need to know the number of times a set of conditions are satisfied in a session, you can create a field and assign a value 1 to it to indicate the condition was satisfied for the event. This gives you the capability to implement the abnormal number based use cases.

A rule can use the field to know the unique number of processes that satisfied the condition.

```
DistinctCount(win_critical_command, 'process-created')
```

5.2.2. CONTEXT RETRIEVAL

Many of the event enrichers perform context enrichment. Enrichers that do this use a parsed field value as a key to a context table to extract the value into a new field.

```
user-email {...
      Map = [
        {
          Field = "user"
          Value = """GetValue('email_user',toLower(user_email))"""
        }...
```

In the above example the 'user' field is created from the user_email context table. When only the 'user_email' is parsed from the log, you can fetch from this context table the AD 'user' value mapped to the 'user_email' which will stitch the event to the user timeline.

5.2.3. NEW FIELD BASED ON INFORMATION WITHIN THE EVENT

Most of the time in enrichment you can take fields from the message to create new fields based on some conditions using logical expressions.

Example 1:

```
local-user {
      EventTypes =
['batch-logon','file-delete','file-read','file-write','privileged-
access','privileged-object-access','process-created','service-
logon','workstation-locked','workstation-unlocked','local-logon','remote-
access','remote-logon','account-password-change','account-password-
reset','account-lockout','account-unlocked','account-enabled','account-
disabled','account-deleted','account-creation','member-added','member-removed']
      Condition = "exists(domain) && ((exists(dest_host) && dest_host = domain)
|| InList(toLower(domain),'workgroup', 'window manager', 'font driver host'))
&& vendor='Microsoft Windows' && !
InList(event_code,'4648','4769','673','676','552') and not EndsWith(user, '$')
and !InList(toLower(user),'system','local service','network service','anonymous
logon')"
      Map = [
        {
          Field = "user_type"
          Value = """'local'"""
        },
        {
          Field = "user"
          Value = "concat(user, ' (', dest_host, ')')"
        }
      ]
    }
```

In the above example a `user_type` field is created, which holds an attribute about the user, such as whether the user is a local user.

Example 2:

```
security-alert-local_asset {
      EventTypes =
['security-alert','dlp-alert','process-alert','network-alert','database-alert']
      Condition = "exists(src_host) || exists(src_ip) || exists(dest_host) ||
exists(dest_ip)"
      Map = [
        {
          Field = "local_asset"
          Value =
"""if(isSiteLocal(src_ip),first(src_host,src_ip),if(isSiteLocal(dest_ip),first(d
est_host,dest_ip),first(src_host,src_ip,dest_host,dest_ip)))"""
        }
      ]
    }
```

In the above example, we look at alert based event types and determine the field to be made the local asset based on priority and isSiteLocal() function.

5.2.4. FIELD MODIFICATION

In the following example, we modify an existing field to create a field that can be used for detection by existing Advanced Analytics content:

```
bytes-domain {
      EventTypes =
['dlp-email-alert-out','dlp-email-alert-out-failed','dlp-alert','usb-
insert','usb-write','usb-read','dlp-email-alert-in','share-access','print-
activity','file-write','file-delete']
      Condition = "exists(bytes_unit) && !exists(bytes)"
      Map = [
        {
          Field = "bytes_num"
          Value = """replaceAll(bytes_num, ",","")"""
        },
        {
          Field = "bytes"
          Value =
"""Multiply(bytes_num,ReturnIf(ToLower(bytes_unit)='kb',1024,ReturnIf(ToLower(by
tes_unit)='mb',1048576,ReturnIf(ToLower(bytes_unit)='gb',1073741824,0))))"""
        }
      ]
    }
```

Advanced Analytics content related to data transfer sizes operate using bytes (not kilobytes, megabytes, or gigabytes). So, when we parse a value from a log that is not in bytes representation, we modify the parsed value accordingly, multiplying the parsed bytes value (`bytes_num`) by 1024 when we see the `bytes_unit` value is kilobytes, multiplying it by 1024*1024=1048576 when the `bytes_unit` value is megabytes, and so on.

5.2.5. REQUIRED MODEL/RULE FIELD

When you want to accurately track how often a specific activity has occurred, you can track the count pairs of field values. You can also use this information to determine if different field values are better stored in a single field value.

Many enrichers use the 'concat' logical expression to put two field values together.

```
unix-target-id {
      EventTypes =
['account-deleted','account-password-change','account-password-reset']
      Condition = """!exists(target_user) && exists(target_user_id) &&
exists(dest_host) && vendor='Unix'"""
```

```
      Map = [
        {
          Field = "target_user"
          Value = "ReturnIf(target_user_id = '0', concat('root (', dest_host,
')'), concat(target_user_id, ' (', dest_host, ')'))"
        }
      ]
    }
```

In the above example, we create a field called `target_user` that either begins with 'root' if `target_user_id = '0'` or begins with the actual `target_user_id` field if the value is not equal to 0.

```
netflow-scanhost {
      EventTypes = ['netflow-connection']
      Condition = "exists(src_host)"
      Map = [
        {
          Field = "src_host_time"
          Value = """concat(src_host, '-', take(time,9))"""
        }
      ]
    }
```

In the above example we concatenate `src_host` and the time field. We can concentrate with time to track whether multiple events relating to a certain activity happened within a single second.

A rule can now use:

```
"""DistinctCountByIf(dest_host, src_host_time, src_locality = 'internal',
'netflow-connection') = 20"""
```

to track whether a host (`src_host`) reached out to another host (`dest_host`) 20 times within a second. This works because the time value is concatenated to the `src_host` value, and because `src_host` will not change, we can rely on that if the entire field does not change, then the time is the same for the different events.

5.3. Event Enricher Configurations

Here is the syntax for an event enricher:

```
EventTypes = ["event_type"]         ### what event types should be evaluated
against this enricher. Using [] means the enricher will apply to all event
types.

Condition = "exists(certain_field) OR endsWith(some_field, '.bat')" .
    ### conditions based on Exabeam's logical expressions

Map = [ { field = "new_field", value = "anything I want" }, { field =
"new_field_2", value = """'100'""" } ]         ### new field definitions
```

- First, restrict what events are enriched, and then define the fields and values to be created.
- Restriction is done using the 'EventType' and 'Condition' field.
- Make sure you only look at events that are of the type(s) specified in the 'EventTypes' field, expressed as an array of `event_types`.
- Further restrict what gets enriched by adding the analytics engine expressions in the conditions parameter. In the above example, we only enrich the event if the field 'some_field' ends with '.bat' or we enrich the event if the field 'certain_field' already exists.

5.4. Additional Enrichment Guidelines

Within an enricher, you can enrich multiple fields.

You can even enrich on a field created higher up in the enricher.

6. Exabeam Persistence and Templates

Persistence refers to saving the existence of a field which has been parsed/enriched in the Mongo database so that it can be used mainly by the Exabeam Advanced Analytics Restful Web Services for display purposes on the UI.

Persisting fields are mandatory for displaying fields associated with events on UI.

6.1. Persistence Definition

The default config for persistence can be found in the `content_default.conf` file (path: `martini/config/default` directory) under the sections `RequiredPersistedEventFields` and `PersistedEventFields`.

`RequiredPersistedEventFields` specifies the fields that need to be persisted for every event, while `PersistedEventFields` specifies for each event type the fields that need to be persisted to the database in addition to those listed under `RequiredPersistedEventFields`.

6.1.1. PERSIST A FIELD

If you want to persist a new field associated with an event on the UI, you need to add the new field entry in `PersistedEventFields` section. For example, if you want to persist a new field "vpn_source_location" associated with vpn-login event which you parsed/enriched, then it has to be added as shown below to `PersistedEventFields`.

NOTE

Make sure you define this config in the custom config file (`/opt/exabeam/config/custom/custom_exabeam_config.conf`), and not by changing the default file.

```
PersistedEventFields {
----------------------
----------------------
vpn-login = [_id,
  vendor,
  src_ip,
  src_host,
  "GetValue('country_code',src_ip)",
  "GetValue('isp',src_ip)",
  "GetValue('zone_info',dest)",
  src_translated_ip,
  dest_host,
  dest_ip,
  src_network_type,
  realm,
  os,
  vpn_source_location]
----------------------
----------------------
}
```

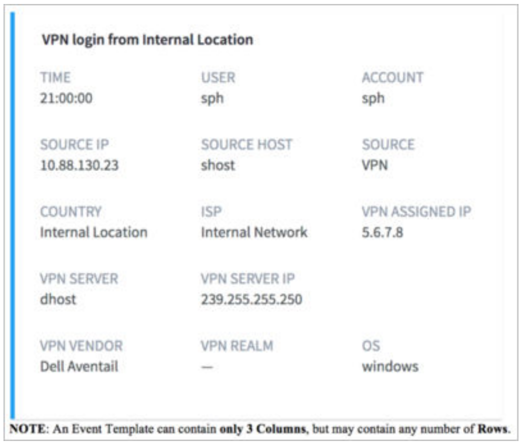
6.1.2. WHEN TO USE PERSISTENCE

You should use persistence to display fields associated with events on the UI.

Suppose you want to display a new field, say `vpn_source_location` associated with an event type vpn-login on the UI, you need to add the field in Event Template associated with vpn-login event.

6.2. Event Template

Event templates are used to display fields associated with an event in the UI. The below shows the event template `VpnLoginTemplate`, which is used for displaying fields associated with vpn-login in the UI. The below template is used to display fields such as time, user, account, src_ip, src_host, source, getvalue ('country_code', src_ip), getvalue('isp', src_ip), src_translated_ip, dest_host, dest_ip, vendor, realm, and os associated with vpn-login event on the UI.



VPN login from Internal Location

TIME 21:00:00	USER sph	ACCOUNT sph
SOURCE IP 10.88.130.23	SOURCE HOST shost	SOURCE VPN
COUNTRY Internal Location	ISP Internal Network	VPN ASSIGNED IP 5.6.7.8
VPN SERVER dhost	VPN SERVER IP 239.255.255.250	
VPN VENDOR Dell Aventail	VPN REALM —	OS windows

NOTE: An Event Template can contain **only 3 Columns**, but may contain any number of **Rows**.

```
VpnLoginTemplate {
    rows = [
        {
        columns = [
            {
            label = "TIME"
            value = "time|event.time"
            },
            {
            label = "USER"
            value = "user|event.user"
            },
            {
            label = "ACCOUNT"
            value = "user|event.account"
            icon = "AccountSwitch"
            }
        ]
        },
        {
        columns = [
            {
            label = "SOURCE IP"
            value = "asset|event.src_ip"
            },
            {
            label = "SOURCE HOST"
            value = "asset|event.src_host"
            },
            {
            label = "SOURCE"
            value = "default|event.source"
            }
        ]
        },
        {
        columns = [
            {
            label = "COUNTRY"
            value = "location.country|event.getvalue('country_code', src_ip)"
            },
            {
            label = "ISP"
            value = "location.isp|event.getvalue('isp', src_ip)"
            },
            {
            label = "VPN ASSIGNED IP"
            value = "default|event.src_translated_ip"
            }
        ]
        },
        {
        columns = [
            {
            label = "VPN SERVER"
            value = "default|event.dest_host"
            },
            {
            label = "VPN SERVER IP"
            value = "default|event.dest_ip"
            }
        ]
        },
        {
        columns = [
            {
            label = "VPN VENDOR"
            value = "default|event.vendor"
            },
            {
            label = "VPN REALM"
            value = "default|event.realm"
            },
            {
```

```
            label = "OS"
            value = "default|event.os"
            }
        ]
        }
    ]
}
```

6.3. Event Templates Definition

The default config for event templates can be found in `event_templates_default.conf` file (path: `tequila/conf/default/` directory). Every event type has an associated template defined and you can find the name of the associated template under the `DetailsTemplate` parameter entry in `EventFormats`.

Therefore, if you want the template defined for vpn-login event, you need to search for an entry for vpn-login in `EventFormats`. You will find the below entry, where the template name `VpnLoginTemplate` is associated with the `DetailsTemplate` parameter.

```
EventFormats {
  -------------------------
  -------------------------
  vpn-login {
    DisplayName = "VPN login"
    Description = "Remote access VPN login attempt either from a public IP
address or from an internal network address was successful."
    HeaderTemplate = "VPN login from {location.country|
event.getvalue('country_code', src_ip)}"
    DetailsTemplate = "VpnLoginTemplate"
  }
  -------------------------
  -------------------------
}
```

Then, you need to search for the template name `VpnLoginTemplate` entry, under Templates, to get the Event Template for vpn-login event as shown below:

```
Templates {
 -----------------------------
 -----------------------------
VpnLoginTemplate {
  rows = [
    {
      columns = [
        {
          label = "TIME"
          value = "time|event.time"
        },
        {
           label = "USER"
           value = "user|event.user"
        },
        {
           label = "ACCOUNT"
           value = "user|event.account"
           icon = "AccountSwitch"
        }
      ]
  },
  {
     columns = [
     {
        label = -------
        value = ------
     }
   ]
  },
  --------------------
  --------------------
}
----------------------
----------------------
}
```

6.3.1. ADD A FIELD IN AN EVENT TEMPLATE

Let's take an example of adding a new field named `vpn_source_location` associated with the vpn-login event in its event template. At first, you search for an entry vpn-login in `EventFormats` in the default config file and you will get an entry for vpn-login, where you then look for `DetailsTemplate` parameter to find the template name `VpnLoginTemplate`. Then, you search for `VpnLoginTemplate` entry under Templates to fetch the event template config as shown in Event Templates Definition.

NOTE

Make sure you define this config in the custom config file (`/opt/exabeam/config/custom/custom_exabeam_config.conf`), and not by changing the default file.

There can only be three columns (one field per column) with respect to each row for display purposes. So, if you want to add a new field entry, and there are already three subsections or fields under `columns = [ section ]`, you cannot add a new entry in that particular row. In that case, you need to add a new `columns = [ section ]` and add an entry. Also, if you see there are only two subsections under `columns = [ section ]`, then you can add your field under that section itself. Let's consider adding `vpn_source_location` field, and please refer to the below template which illustrates adding this entry into the template:

```
Templates {
 -----------------------------
 -----------------------------
VpnLoginTemplate {
  rows = [
    {
      columns = [
        {
          label = "TIME"
          value = "time|event.time"
        },
        {
           label = "USER"
           value = "user|event.user"
        },
        {
           label = "ACCOUNT"
           value = "user|event.account"
           icon = "AccountSwitch"
        }
      ]
  },
  --------------------
  --------------------
  {
      columns = [
        {
          label = "VPN SERVER"
          value = "default|event.dest_host"
        },
        {
          label = "VPN SERVER IP"
          value = "default|event.dest_ip"
        },
        {
          label = "VPN SRC LOCATION"
          value = "default|event.vpn_source_location"
        }
           ]
  },
  --------------------
  --------------------
}
----------------------
----------------------
}
```

As shown in the above case, `vpn_source_location` was added to `columns = [ section ]` in which only two entries existed, and which there was an option to add a third entry for a new field.

Please note that the parameter label defines the name of the field displayed on the UI and value parameter should contain the field for which you need the value to be displayed. Most importantly, the field which you want to display has to be persisted as described in the earlier section. If not, you will not be able to display the value for your field. In this case, `vpn_source_location` has to be persisted in Mongo, and then added to the template in order to display it with respect to the vpn-login event.

6.4. Custom Persistence and Templates

In addition to default config parameters. You can update an event template with a custom field by persisting it in MongoDB. Then you need to make changes as discussed below to custom files.

- If it is persistence, and you want to persist custom field `vpn_source_location` in MongoDB, then add the below section in your custom config file (`/opt/exabeam/config/custom/custom_exabeam_config.conf`):

```
PersistedEventFields {
            vpn-login = [_id,
            vendor,
            src_ip,
            src_host,
            "GetValue('country_code',src_ip)",
            "GetValue('isp',src_ip)",
            "GetValue('zone_info',dest)",
            src_translated_ip,
            dest_host,
            dest_ip,
            src_network_type,
            realm,
            os,
            vpn_source_location]
          }
```

NOTE

Do not forget to enclose the entry with `PersistedEventFields { }` as shown above in the custom config file.

- For event templates, and you want to display custom field (`vpn_source_location`) value on the UI, then add the below section in your custom config file (`/opt/exabeam/config/tequila/custom/event_templates.conf`):

```
Templates {
           VpnLoginTemplate {
          rows = [
          {
               columns = [
               {
                   label = "TIME"
                   value = "time|event.time"
               },
               {
                   label = "USER"
                   value = "user|event.user"
               },
               {
                   label = "ACCOUNT"
                   value = "user|event.account"
                   icon = "AccountSwitch"
               }
             ]
          },
          --------------------
          --------------------
          {
               columns = [
               {
                   label = "VPN SERVER"
                   value = "default|event.dest_host"
               },
               {
                   label = "VPN SERVER IP"
                   value = "default|event.dest_ip"
               },
               {
                   label = "VPN SRC LOCATION"
                   value = "default|event.vpn_source_location"
               }
                  ]
          },
          --------------------
          --------------------
       }
     }
```

NOTE
Do not forget to enclose the entry with `Templates { }` as shown above in the custom config file.

6.5. Restart Services for Persistence and Templates

- If you make any updates for persistence and you want to see if a new field value is persisted on MongoDB, you need to restart the analytics engine (`exabeam-analytics-stop;exabeam-analytics-start`).
- If you make any updates for event templates and you want to see the changes on UI, you need to restart the web components (`web-stop;web-start`).
- If you make updates for persistence and want them displayed on the UI, then you also make updates to event templates, you need to restart both the analytics engine as well as the web components.

NOTE
It is mandatory for a field to be persisted if it is required to display it on the UI using templates.

7. Exabeam Models

Exabeam Advanced Analytics performs anomaly detection using models. Without models, rules can only score on 'fact' based logic, the kind that looks for specific things in the logs or counting for specific values over an entire session. Models also track historical values (features) for a given item (scope). For example, tracking hosts (feature values) a user (scope) has logged into. If the current value is deemed to be abnormal, versus the historical values in the model, a rule can associate a score with this anomaly. Anomaly detection is performed by calculating a number of statistics about the features in a given model to check whether the feature value seen, in an event being evaluated, is unusual or not.

NOTE

The Advanced Analytics pipeline is as follows:

parsing > event building > enrichment > session building > **modeling** > rule triggering

Advanced Analytics statistical profiling is not only about user-level data. In fact, Exabeam profiles other entities, including hosts and peer groups. RAM and performance permitting, just about anything can be modeled. If it is parsed, then the parsed/enriched field can be used as either the scope or the feature in a model. Ensuring how large a model might grow as well as understanding what values in the future may populate the model and how it will affect anomaly detection are factors to consider when deciding what to make a scope or feature for a model.

7.1. Types of Models

There are three types of models:

- **"CATEGORICAL"** – As the name suggests, this type of model is used to train on values that are strings such as host or user names.
- **"NUMERICAL_CLUSTERED"** – This type of model is used to train on numerical values such as the number of hosts a user logs into a session.
- **"NUMERICAL_TIME_OF_WEEK"** – This type of model is used to train on the time when events occur.

7.1.1. CATEGORICAL MODELS

Let's compare the modeling endpoint entity-like processes in both UBA and EA perspectives.

UBA (user based) model

```
EPA-HP {
  ModelTemplate = "Processes for the user"
  Description = "Models processes for this user"
  Category = "End Point Activity"
  IconName = ""
  ScopeType = "USER"
  Scope = """user"""
  Feature = """process_name"""
  FeatureName = "process"
  FeatureType = "process_name"
  TrainIf = """sequenceCount(process_name,'process-created','process-
alert')=1"""
```

```
  ModelType = "CATEGORICAL"
  AgingWindow = "32"
  CutOff = "10"
  Alpha = "2"
  MaxNumberOfBins = "10000000"
  ConvergenceFilter = "confidence_factor&gt;=0.8"
  HistogramEventTypes = [   "process-created",   "process-alert"
  ]
  Disabled = "FALSE"
}
```

- `EPA-HP` models all the process names for this user. This is evident by inspecting Scope and Feature values. Since we are modeling process names, the `ModelType` is `CATEGORICAL`.
- `Scope` is `user`, a parsed field in process-created/process-alert events.
- `Feature` is `process_name`, a parsed field which means name of process seen in process-created/process-alert events.
- `Category` is `End Point Activity` as process related activity is categorized as endpoint activity.
- `sequenceCount(process_name,'process-created','process-alert')=1` expression makes sure that the model trains when it notices different values of `process_name` for the user in process-created/process-alert events.
- Expressions in such models generally use `Count/sequenceCount/DistinctCount/sequenceDistinctCount`.
- Histogram for this model displays process names on a host in a specific range of time.

EA (asset based) model

```
A-EPA-HP {
  ModelTemplate = "Processes on this asset"
  Description = "Models processes on this asset"
  Category = "End Point Activity"
  IconName = ""
  ScopeType = "DEVICE"
  Scope = """dest_host"""
  Feature = """process_name"""
  FeatureName = "process"
  FeatureType = "process_name"
  TrainIf = """CountBy(process_name,dest_host,'process-created','process-
alert','process-network')=1"""
  ModelType = "CATEGORICAL"
  AgingWindow = ""
  CutOff = "10"
  Alpha = "3"
  MaxNumberOfBins = "5000000"
  ConvergenceFilter = "confidence_factor&gt;=0.8"
  HistogramEventTypes = [    "process-created",    "process-alert",    "process-
network"
  ]  SequenceTypes = [asset]
  Disabled = "FALSE"
}
```

- `SequenceTypes = [asset]` and Model IDA-EPA-HP (which starts with A-) specifies that this Model is Asset (EA) based.
- `A-EPA-HP` models all the process names on an asset (asset is `dest_host` in this case). This is evident by inspecting `Scope` and `Feature` values. Since we are modeling process names, the `Type` is `CATEGORICAL`.
- `Scope` is `dest_host` (asset), a parsed field which means destination host on which process-created/process-alert/process-network events have taken place.
- `Feature` is `process_name`, a parsed field which means name of process seen in process-created/process-alert/process-network events.
- `Category` is `End Point Activity` as process related activity is categorized as endpoint activity.
- `CountBy(process_name,dest_host,'process-created','process-alert','process-network')=1` expression makes sure that the model trains when it notices different values of `process_name` with regard to `dest_host` in process-created/process-alert/process-network events.
- Expressions in such models generally use `CountBy/CountByIf/DistinctCountBy/DistinctCountByIf`.
- Histogram for this model displays process names on a host in a specific range of time.

7.1.2. NUMERICAL CLUSTERED MODELS

Let's compare the modeling amount of data uploaded to web per day in both UBA and EA perspectives.

UBA (user based) model

```
WEB-UBytesSum-Out {
  ModelTemplate = "Sum of bytes written/uploaded to the web in a day by the
user"
  Description = "Models the amount of data (in bytes) that were uploaded to the
web in a day by the user"
  Category = "Web Activity"
  IconName = ""
  ScopeType = "USER"
  Scope = "user"
  Feature = "sequenceSum(bytes_in_post,'web-activity-allowed')"
  FeatureName = "bytes"
  FeatureType = "quantity"
  TrainIf = """sequenceSum(bytes_in_post,'web-activity-allowed')&gt;0"""
  ModelType = "NUMERICAL_CLUSTERED"
  BinWidth = "5"
  AgingWindow = ""
  CutOff = "10"
  Alpha = "1"
  ConvergenceFilter = "confidence_factor&gt;=0.8"  HistogramEventTypes = [
    "sequence-end"
  ]
  Disabled = "FALSE"
}
```

- `WEB-UBytesSum-Out` models amount of data (in bytes) uploaded to web per day by the user. Since we are modeling quantity of data, the `ModelType` is `NUMERICAL_CLUSTERED`.
- `Scope` is `user`, a parsed field in web-activity-allowed events.
- `Feature` is `sequenceSum(bytes_in_post,'web-activity-allowed')`, where bytes_in_post is an enriched field which makes sure only bytes uploaded are tracked in web-activity-allowed events.
- `sequenceSum(bytes_in_post,'web-activity-allowed')>0` expression makes sure that the model trains when the sum of bytes uploaded to web sequence by user > 0 in web-activity-allowed events.
- `sequence-end` events mentioned in `HistogramEventTypes` signifies that the histogram for this model is generated at the end of sequence.
- Expressions in such models generally use `sum/sequenceSum/DistinctCount/sequenceDistinctCount`.
- Histogram for this model displays amount of data (in bytes) uploaded to web per day by the user in a specific range of time.

EA (asset based) model

```
A-WEB-BytesSum-Out {
  ModelTemplate = "Sum of bytes written/uploaded to the web in a day by the
asset"
  Description = "Models the amount of data (in bytes) that were uploaded to the
web in a day by the asset"
  Category = "Web Activity"
  IconName = ""
  ScopeType = "DEVICE"
  Scope = "src_host"
  Feature = "sumBy(bytes_in_post,src_host,'web-activity-allowed')"
  FeatureName = "bytes"
  FeatureType = "quantity"
  TrainIf = """sumBy(bytes_in_post,'web-activity-allowed')&gt;0"""
  ModelType = "NUMERICAL_CLUSTERED"
  BinWidth = "5"
  AgingWindow = ""
  CutOff = "10"
  Alpha = "1"
  ConvergenceFilter = "confidence_factor&gt;=0.8"
  HistogramEventTypes = [    "sequence-end"
  ]
  SequenceTypes = [asset]
  Disabled = "FALSE"
}
```

- `SequenceTypes = [asset]` and Model IDA-WEB-BytesSum-Out (which starts with A-) specifies that this Model is Asset (EA) based.
- `A-WEB-BytesSum-Out` models amount of data (in bytes) uploaded to web per day by the asset (asset is `src_host` in this case). Since we are modeling quantity of data, the `ModelType` is `NUMERICAL_CLUSTERED`.
- `Scope` is `src_host` (asset), a parsed field which means source host which has uploaded data in web-activity-allowed events.
- `Feature` is `sumBy(bytes_in_post,src_host,'web-activity-allowed')` where bytes_in_post is an enriched field which makes sure only bytes uploaded are tracked in web-activity-allowed events.
- `sumBy(bytes_in_post,'web-activity-allowed')>0` expression makes sure that the model trains when the sum of bytes uploaded to web sequence by `src_host` > 0 in web-activity-allowed events.
- `sequence-end` events mentioned in `HistogramEventTypes` signifies that the histogram for this model is generated at the end of sequence.
- Expressions in such models generally use `sumBy/sumByIf/DistinctCountBy/DistinctCountByIf`.
- Histogram for this model displays amount of data (in bytes) uploaded to web per day by the `src_host` (asset) in a specific range of time.

7.1.3. NUMERICAL TIME OF WEEK MODELS

Let's have a look at modeling time of print activity for a user.

UBA (User based) Model

```
PR-UT-TOW {
  ModelTemplate = "Print activity time for user"
  Description = "Models the times of day that this user performs print activity"
  Category = "Print Activity"
  IconName = "user"
  ScopeType = "USER"
  Scope = "user"
  Feature = "TimeOfWeek()"
  FeatureName = "Time"
  FeatureType = "Time"
  TrainIf = """TRUE"""
  ModelType = "NUMERICAL_TIME_OF_WEEK"
  AgingWindow = ""
  CutOff = "10"
  Alpha = "1"
  ConvergenceFilter = "confidence_factor>=0.8"
  HistogramEventTypes = [
    "print-activity"
  ]
  Disabled = "FALSE"}
 // End of PR-UT-TOW
```

UBA (User based) Model

- `PR-UT-TOW` models the time at which print activity took place for the user. Since, we are modeling time, the `ModelType` is `NUMERICAL_TIME_OF_WEEK`.
- `Scope` is `user`, parsed field in print-activity events.
- `Feature` is `TimeOfWeek()` which fetches the day of the week of the event.time as (0..6) including fractions.
- Expressions in such models generally use `count/sequenceCount`.
- Histogram for this model displays print activity times by the user in a specific range of time.

7.2. Model Categories

The following are some model categories used in default content:

- Alerts
- Applications
- Asset Activity Monitoring
- Assets
- Critical Server Login
- DLP
- Database Activity
- Devices
- Directory Service
- Domain Controller Login
- Email
- End Point Activity
- File Access
- Groups
- Identities
- Locations
- Network
- Network Alert
- Other
- Physical Access
- Print Activity
- Privilege Access
- Time
- TopGroups

- TopUsers
- Users
- VPN
- Web Activity
- Windows Audit Change
- Workstations
- Zones

8. Exabeam Rules

Once a log has passed the event building and enrichment phase it is now ready to be processed against the "risk engine", where it will get evaluated against a set of rules that automatically ship with Exabeam.

> **NOTE**
> The Advanced Analytics pipeline is as follows:
>
> parsing > event building > enrichment > session building > modeling > **rule triggering**

Rules contain the logical expressions that define unwanted and malicious behavior (or behavior you want to be alerted on). They provide scoring to the timeline and all the values in the UI.

Every single definition that recognizes a specific malicious behavior that you would like to add points to a timeline are defined in a Rule.

Rules are mainly defined by a logical expression, set in the `RuleExpression` field, and when this expression evaluates to true, the rule "triggers" and points are added to the relevant session or sequence. For example,

```
"""InList(process_name, 'evil.exe', 'ransomware.exe')"""
```

8.1. Types of Rules

There are two types of rules:

8.1.1. FACT BASED RULES

These rules use only the field values from the current event in order to determine whether to trigger or not. This is opposed to relying on historical data in models. Think of these as your simple correlation rules. If x is seen in field y, trigger rule.

8.1.2. MODEL BASED RULES

These rules use historical information stored in models. The rules typically trigger when the event being evaluated is considered 'anomalous' within the context of the model. Exabeam, by default, ships a large number of rules that are separated into different `rules_*.conf` files based on the type of malicious behavior they trigger on. For example, rules relating to web activity will be in the `rules_webactivity.conf` file. We have a file dedicated to rules related to web activity. This is done for organization, and rules can be placed in any referenced rule file.

8.2. Create Rules

Rules are meant to define some logical activity (for examples, a vpn login) you want to be alerted on. Any event that successfully goes into a timeline can trigger a rule and add points to a timeline and elevate a user to notable status. So, rules indicate the flagging of bad, suspicious, and benign behavior.

Key Rule Ingredients:

- **RuleID** - The string that is the 'name' of the HOCON block in the rule configuration file. If another rule seen later (below) in the `rules.conf` file has the same key, the first rule configuration will be thrown out.

- **ClassifyIf** - Supports the analytics engine expressions that are usable in the RuleExpression, meaning this is an additional place to place rule expression logic, though generally the logic put in this attribute usually identifies restrictions and scoping of when the rule should trigger. Many rules use this field to place a session/sequence "count" analytics engine expression that limits the rule from triggering more than once for a specific value.

> **NOTE**
> For fact based rules, set ClassifyIf to "TRUE".

- **RuleEventTypes** - Events of the event type expressed in this field will be evaluated against the rule. A rule with event type web-activity-denied will never be evaluated against a network-connection-denied event.
- **RuleExpression** - Rule logic. When should this rule 'fire'. For example, `RuleExpression ="""  process_name="malicious.exe" """`.
- **RuleType** - session, asset, endpoint, web, file, database, external or account-lockout. All events in Advanced Analytics fall under one of these seven event buckets. For example, process-created events are only put into the 'endpoint' sequence. So, even if the `RuleEventType` contains process-created, if `RuleType = 'session'`, this rule will never fire for process-created events.

When a rule is found to not trigger when it should, it is normally one of these five fields that are edited to fix the rule.

When testing new rules, take the events you want to trigger against, and ensure that the event type is included in `RuleEvent` types, and that the event type is included in the `ruleType` (session, asset, endpoint sequence, etc.).

8.3. Rule Dependency and Chaining

Rules can have relationships with other rules through the use of the `DependencyExpression` attribute and `WasRuleFired`.

Complex sets of rules can be created using the 'Dependency Expression'. This expression checks for other triggered rules that triggered for the same event. That is, if an event triggers Rule_A, and Rule_B has `"dependencyExpression='Rule_A' "` then Rule_B would trigger only if rule A has triggered (in addition to its own expression). Dependency expressions can use "and", "or", and "not" to define complex dependencies.

The `WasRuleFired` expression is used to determine if a specific rule has previously triggered in the session or sequence and optionally on which values. If rule_X has `WasRuleFired('Rule_Z')` in its `RuleExpression`, it will trigger only if 'Rule_Z' was previously triggered. The expression `WasRuleFired('Rule_Z', dest_host)` indicates that the current rule should trigger only if Rule_Z has triggered previously in the session/sequence and the value of the `dest_host` in the event it triggered on is the same as that value in the current event. `WasRuleFired` can be used also to negate: `!WasRuleFired('Rule_X')` which means trigger the current rule only if the specified rule has not triggered. This expression is often used to ensure a certain rule triggers only once per session/ sequence.

8.4. Internal Rules

Internal rules are rules with a score of 0. They are not displayed in the UI and are used by other rules as dependencies. They are used to identify events that have no security value (and therefore no score), so they identify a situation that could be significant for another rule. For example, the internal NEW_USER rule is used to identify a new user in the environment. It is used as a dependency for different rules that identify access to privileged machines, executives' machines, etc. Together they identify a situation in which a new user is accessing a privileged machine. If we want to change how we identify a new user, it has to be done only in the NEW_USER rule and will propagate to all other rules.

Another use of internal rules is to condition on information in more than one model. Since a single rule can condition on data in only one model, multiple internal rules can condition on data in different models and a single score giving rule would use these as dependencies to provide the score if all conditions materialize.

8.5. Additional Rule Guidelines

Listed below is a list of additional guidelines and features.

- Triggered rule info is searchable in the 'triggered_rule_db' in Mongo.
- RuleExpressions can incorporate any parsed field into the logic. For asset based rules, if you want to use a parsed field in a 'countby' expression, that parsed field must be persisted.
  - When a Model-Based-Asset-Rule uses `CountBy(field_1, field_2, event_types)`, both `field_1` and `field_2` must be persisted for that event type in the `PersistedEventFields` definition in the enricher `content_default.conf` file.
- User based rules use `Count`, `SequenceCount`, and `DistinctCount` for gathering session/sequence data.
- Asset based rules use `CountBy` for all purposes of gathering sequence data. All asset events are 'sequence' events, and thus `CountBy` can be used for gathering sequence data for any event type.

8.6. Fact Based Rules

On the left is a 'User' rule, on the right an 'Asset' rule. The analytics engine needs this distinction `RuleType = "asset"` to score only against assets as well as an aggregation expression.

<table>
<tr><th>UBA (User Based) Rule</th><th>EA (Asset Based) Rule</th></tr>
<tr><td>

```
FA-Outlook-pst {   RuleName = "A file ends with
either pst or ost"   RuleDescription = "A file
copied ends with either pst or ost"
  ReasonTemplate = "PST/OST file copied"
  AggregateReasonTemplate = "PST/OST file copied"
  RuleType = "file"   RuleCategory = "File
Activity"   ClassifyIf = """TRUE"""
  RuleEventTypes = [     "file-write"   ]
  Disabled = "FALSE"   Model = "FACT"
  FactFeatureName = "src_file_name"   Score =
"20.0"   RuleLabels {   mitre = ["T1114"]   }
  PercentileThreshold = "0.1"   RuleExpression =
"""sequenceCount(src_file_name,'file-write')=1 &&
(endsWith(toLower(src_file_name), '.pst') ||
endsWith(toLower(src_file_name), '.ost'))"""
  DependencyExpression = "FA-Outlook" }
```

</td><td>

```
A-ALERT-DISTINCT-NAMES {
  RuleName = "Various security alerts on asset"
  RuleDescription = "At least three distinct
security alerts were reported for the asset.
This raises the probability that the asset is
compromised."
  ReasonTemplate = "Third distinct security
alert on asset"
  AggregateReasonTemplate = ""
  RuleType = "asset"
  RuleCategory = "Security Alert"
  ClassifyIf = """TRUE"""
  RuleEventTypes = [
    "security-alert"
  ]
  Disabled = "FALSE"
  Model = "FACT"
  FactFeatureName = """src_host"""
  Score = "25.0"
  RuleLabels {
    mitre = ["T1066"]
  }
  PercentileThreshold = "0.1"
  RuleExpression =
"""DistinctCountBy(alert_name, asset,
'security-alert')=3 && !WasRuleFired('A-ALERT-
DISTINCT-NAMES')"""
  DependencyExpression = "NA"
  Aggregation {
    DataExpr = """DistinctCountBy(alert_name,
asset, 'security-alert')=3 && !WasRuleFired('A-
ALERT-DISTINCT-NAMES')"""
    EventExpr = "TRUE"
  }
}
```

</td></tr>
</table>

<table>
<tr><th>UBA (User Based) Rule</th><th>EA (Asset Based) Rule</th></tr>
<tr>
<td>• FA-Outlook-pst rule is used to detect suspicious files (like .pst and .ost files) being copied. Since, we are focusing specifically on .pst and .ost files only, this is considered fact based instead of model based, which is indicated by Model = "FACT".
• RuleType = "file" and RuleCategory = "File Activity" indicate that the rule deals with files.
• ClassifyIf = """TRUE""" is mandatory since it is a fact based rule.
• The rule investigates "file-write"events to look whether .pst/.ost file is being copied. This is explained by specifying "file-write"event type in RuleEventTypes parameter.
• FactFeatureName = "src_file_name" describes that the feature value is src_file_name which is a parsed field. Also, this value will be shown while using featureValue in the ReasonTemplate and AggregateReasonTemplate parameters.
• Based on criticality of the rule, we can assign appropriate score for the rule. Here Score = "20.0".
• mitre = ["T1114"] in RuleLabels indicate that this rule can be tagged with the MITRE technique T1114.
• Percentile Threshold = "0.1" means that for the purposes of this rule we only consider events that appear below the 10th percentile to be abnormal.
• sequenceCount(src_file_name,'file-write')=1 makes sure that the rule triggers only for different values of src_file_name in file-write events. (endsWith(toLower(src_file_name), '.pst') || endsWith(toLower(src_file_name), '.ost')) makes sure that the file extension of src_file_name is either .pst or .ost. The && between the expressions in RuleExpression indicates both conditions should be satisfied for the rule to trigger.
• DependencyExpression = "FA-Outlook" indicates that the rule FA-Outlook-pst triggers only if FA-Outlook rule has already been triggered.</td>
<td>• Rule IDA-ALERT-DISTINCT-NAMES (which starts with A-), presence of Aggregation parameter and RuleType = "asset" in Rule config specify that this rule is asset (EA) based.
• A-ALERT-DISTINCT-NAMES rule is used to detect multiple security alerts (3 different alerts in this case) on a src_host which is a parsed field meaning source host (asset). Since, we are focusing specifically on 3 different alerts only, this is considered fact based instead of model based, which is indicated by Model = "FACT".
• RuleCategory = "Security Alert" indicates the the rule deals with security alerts.
• ClassifyIf = """TRUE""" is mandatory since it is a fact based rule.
• The rule investigates "security-alert" events to detect multiple (3 different) security alerts on an asset. This is explained by specifying "security-alert" event type in RuleEventTypes parameter.
• FactFeatureName = "src_host" describes that the feature value is src_host which is a parsed field. Also, this value will be shown while using feature Value in the ReasonTemplate and AggregateReasonTemplate parameters.
• Based on criticality of the rule, we can assign appropriate score for the rule. Here Score = "25.0".
• mitre = ["T1066"] in RuleLabels indicate that this rule can be tagged with the MITRE technique T1066.
• PercentileThreshold = "0.1" means that for the purposes of this rule we only consider events that appear below the 10th percentile to be abnormal.
• DistinctCountBy(alert_name, asset, 'security-alert')=3 makes sure that the rule triggers only if 3 distinct values of security alerts are observed on asset. !WasRuleFired('A-ALERT-DISTINCT- NAMES') makes sure that this rule does not trigger if it has already been triggered. The && between the expressions in RuleExpression indicates both conditions should be satisfied for the rule to trigger.
• DependencyExpression = "NA" indicates that the rule is independent of any other rule.
• Aggregation parameter is mandatory for EA rules. It consists of three different parameters. DataExpr which is used to specify expressions used in triggering the rule (like DistinctCountBy/DistinctCountByIf/sumBy/sumByIf etc.) along with expressions which may involve specific conditions (for example custom conditions like user!='System', bytes='100' etc.), EventExpr which is generally TRUE and ModelExpr which is used to specify expressions used in Models like num_observations=0, ConfidenceFactorAboveOrEqual() etc. Generally ModelExpr = """TRUE""" for FACT based rules.</td>
</tr>
</table>

The rule on the left describes a rule most commonly seen in traditional SIEMs. For the parsed field 'x' in an event, if 'x' = 'some_string', trigger the rule. With the context available to Advanced Analytics, 'Fact' based rules have a little more power than this, as shown by the rule on the right. This rule uses the 'DistinctCount' expression to check, within a session, the number of all the different values seen for a field. A 'DistinctCount' on the field 'dest_host' will return the number of dest hosts seen in logs by a user, which is how many different dest hosts were reached by a user in a session. Using an enriched field 'special_field_count_num' that is set to the number 1, a 'Fact' based rule can use the 'Sum' expression to get a count of how many times a specific value in a parsed field was seen.

For example:

Enricher (defined in a different file, see Enrichment):

```
EventTypes = '
Condition = "!InList(tld_domain, "com", "org", "edu", "uk", "co")
Map = [ field = {"abnormal_tld_count", value = "1"} ]RuleExpression =
"Sum(abnormal_tld_count, 'web-activity-denied')=20"
```

will trigger when a user has been denied 20 times trying to access a website that is not a .com, .org., .edu, .uk. , .co site.

8.7. Model Based Rules

For every event that gets processed by the analytics engine, there exists a classification phase. In the classification phase, the event is evaluated against all existing models. In this phase the event is 'triaged' with corresponding models, and corresponding model data is stored with that event, which is later used by the `RuleExpressions`. In rules that are model based, model calculations the `RuleExpression` uses are created when the event is classified. Thus, when you see model logic in rule expression, such as `percentile_threshold_value`, this value is already predetermined.

8.7.1. UBA (USER BASED)

<table>
<tr><th>Model</th><th>Rule</th></tr>
<tr><td>SA-UA {
  ModelTemplate = "Security alert names
for user"
  Description = "Models security alert
names for the user"
  Category = "Other"
  IconName = ""
  ScopeType = "USER"
  Scope = """user"""
  Feature = """alert_name"""
  FeatureName = "alert_name"
  FeatureType = "alert_name"
  TrainIf =
"""count(alert_name,'security-
alert')=1"""
  ModelType = "CATEGORICAL"
  AgingWindow = "32"
  CutOff = "5"
  Alpha = "0.8"
  MaxNumberOfBins = "1000000"
  ConvergenceFilter =
"confidence_factor>=0.8"
  HistogramEventTypes = [
    "security-alert"
  ]
  Disabled = "FALSE"
}</td><td>SA-UA-F {
  RuleName = "First security alert name for user"
  RuleDescription = "This is the first occurrence of this security alert
name for the user"
  ReasonTemplate = "First security alert with name {default|featureValue|
histogram} for user"
  AggregateReasonTemplate = "First security alert name for user: {default|
featureValue|histogram}"
  RuleType = "session"
  RuleCategory = "Security Alert"
  ClassifyIf = """count(alert_name,'security-alert')=1"""
  RuleEventTypes = ["security-alert"
  ]
  Disabled = "FALSE"
  Model = "SA-UA"
  FactFeatureName = "alert_name"
  Score = "10.0"
  RuleLabels {
    mitre = ["T1078"]
  }
  PercentileThreshold = "0.1"
  RuleExpression = """num_observations=0"""
  DependencyExpression = "NA"
}</td></tr>
</table>

Model	Rule
• SA-UA is used to model Security alert names for user. Refer to Modelsfor details on various model attributes.	• SA-UA-F (which ends with -F) is used to detect First Security alert name for the user. • `Model = "SA-UA"` demonstrates that the rule is model based and it depends on the data trained by model SA-UA. • `RuleType = "session"` indicates that it is a session rule. • `RuleCategory = "Security Alert"` indicates that the rule deals with Security alerts. • `count(alert_name,'security-alert')=1` expression in `ClassifyIf` talks about the frequency of rule trigger. Here, this rule triggers once per `alert_name(parsed field) insecurity-alert` events. • The rule investigates "security-alert" events to detect first security alert name for the user. This is explained by specifying "security-alert" event type in `RuleEventTypes` parameter. • `FactFeatureName = "alert_name"` describes that the feature value is `alert_name` which is a parsed field. Also, this value will be shown while using `featureValue` in the `ReasonTemplate` and `AggregateReasonTemplate` parameters. • Based on criticality of the rule, we can assign appropriate score for the rule. Here `Score = "10.0"`. • `mitre = ["T1078"]` in RuleLabels indicate that the this rule can be tagged with the MITRE technique T1078. • `PercentileThreshold = "0.1"` means that for the purposes of this rule we only consider events that appear below the 10th percentile to be abnormal. • `num_observations` in `RuleExpression` indicates how many times the current value (feature) was observed. 0 means never observed before (first time). • `DependencyExpression = "NA"` indicates that the rule does not depend or is independent of any other rule.

8.7.2. EA (ASSET BASED) RULE

All asset rules use 'countBy' instead of 'count' for all event types.

<table>
<tr><th>Model</th><th>Rule</th></tr>
<tr><td>A-FLSh-Count {
ModelTemplate = "Count of failed logons from host"
Description = "Models the number of failed logons from this asset"
Category = "Assets"
IconName = ""
ScopeType = "DEVICE"
Scope = """src_host"""
Feature = """DistinctCountBy(event_id,src_host,'failed-logon')"""
FeatureName = "activity"
FeatureType = "quantity"
TrainIf = """TRUE"""
ModelType = "NUMERICAL_CLUSTERED"
AgingWindow = ""
CutOff = "5"
Alpha = "1"
MaxNumberOfBins = "1000000"
ConvergenceFilter = "confidence_factor>=0.8"
HistogramEventTypes = ["sequence-end"]
SequenceTypes = [asset]
Disabled = "FALSE"
}</td><td>A-FLSh-Count-Ac {
RuleName = "Abnormal number of failed logons from asset (L)"
RuleDescription = "Extremely abnormal number of failed logons from asset"
ReasonTemplate = "({quantity|featureValue}) failed logons from asset, expected around {quantity|percentileThresholdValue|histogram}"
AggregateReasonTemplate = ""
RuleType = "asset"
RuleCategory = "Failed Logon and Account Lockout"
ClassifyIf = """TRUE"""
RuleEventTypes = ["failed-logon"]
Disabled = "FALSE"
Model = "A-FLSh-Count"
FactFeatureName = "NA"
Score = "40.0"
ScoreTarget = src_host
RuleLabels {
mitre = ["T1110","T1078"]
}
PercentileThreshold = "0.1"
RuleExpression = """num_observations<percentile_threshold_count && ConfidenceFactorAboveOrEqual() && percentile_count_distance>5 && !WasRuleFired('A-FLSh-Count-Ac')"""
DependencyExpression = "NA"
Aggregation {
DataExpr = """!WasRuleFired('A-FLSh-Count-Ac')"""
EventExpr = "TRUE"
ModelExpr = """num_observations<percentile_threshold_count && ConfidenceFactorAboveOrEqual() && percentile_count_distance>5"""
}
}</td></tr>
<tr><td>The above model tracks how many times an asset 'gets' a failed logon event in a day.</td><td>This rule triggers when:
A failed-logon event has occurred, and the number of failed-logons in the sequence so far is now considered abnormal compared to what exists in the model. The percentile_count_distance can be used to vary the amount of abnormality you would like to trigger in. Comparing against a bigger value triggers on a 'more abnormal' event(s).</td></tr>
</table>

Appendix A. Appendix

1. Model and Rule Attributes Definitions

Refer to this appendix to learn more about the model and rule attributes for session events.

The following tables and examples provide definitions of the rule and model attributes for session events.

> **NOTE**
> Expressions and logic are slightly different for Entity Analytics rules and models. Also, expressions are slightly different for user based sequences, such as web and endpoint.

1.1. MODEL ATTRIBUTES

Refer to this table to learn more about model attributes.

Attribute	Definition	Example
Model Id	The hocon object name, which must be unique. Used in rules to reference the model, and can also be used to search in Data Insights.	PR-UP
ModelTemplate	Any string describing the model.	Printers for user
Description	Any string providing more detail about the model.	Models the printers used by this user
Category	Any string that groups this model with others that are similar. Will show up in the UI in Data Insights.	Print Activity
IconName	Icon to display next to the model in the UI. Can be left empty.	user
ScopeType	Type of scope. Options are ORG, USER, PEERS, or DEVICE.	USER
Scope	The entity for which the model is created. In this case, a model instance will be an instance for every value of the user field. For models with scope type ORG, this field should be "org".	user
Feature	The value that should be modeled for the entity. In this case, it's the value of the `printer_name` field.	printer_name
FeatureName	Any string displayed as the header of the feature table when viewing the histogram in the UI.	printer
TrainIf	When the model should be trained. Common expressions are count() and TRUE. `count(myField, 'event1','event2',...)=1` means once per value of the myField field is observed in the specified event types. Counts are reset on new sessions/sequences and are per user. The count() expression is independent of the events that will be considered in the model, which means you do not have to include the event types in the model for the count() expression to use them. **NOTE** You cannot use other expressions within count(), for example `count(concat(f1,f2)...)` is not supported. You will have to create an enriched field with the expression you want to count on and use that. In this example the model will be trained once per printer_name as appears in the print-activity events per user per session. This can be combined with other expressions. Setting the field to TRUE would train the model on every event.	count(printer_name,'print-activity')=1

Attribute	Definition	Example
ModelType	Whether the model holds categorical or numerical data. Options are: CATEGORICAL, NUMERICAL_CLUSTERED, or NUMERICAL_TIME_OF_WEEK.	CATEGORICAL
AgingWindow	Starting in Advanced Analytics I48, represents the number of weeks data will be held in the model before purging. The default value is 16.	24
CutOff	Number of events below which the `confidence_factor` or `ConfidenceFactorAboveOrEqual()` will return 0 regardless of actual calculation.	5
MaxNumberOfBins	Starting in Advanced Analytics I48, represents the max number of bins the model is allowed before the instance is disabled.	1000
Alpha	Used in the `confidence_factor calculation`: ((N-C)/N)^a in which N=total data points in the model, C=number of bins, and a=alpha. The higher the alpha the more data would be required for the model to converge.	1

ConvergenceFilter	Used only for internal stats. This parameter signifies the amount of data needed for the model to train. It's calculated using the expression mentioned above. So, in this example, the model does not train the data if it does not satisfy the condition confidence_factor>=0.8.
	The confidence factor (cf) goes from 0 when we have no confidence to 1 when we have full confidence. The formula is
	`cf = [(N-C) / N]α`
	`N: number of observed events.`
	`C: number of unique observed events.`
	`α a factor determining how quickly the confidence grows. The higher the number the slower confidence grows.`
		confidence_factor>=0.8

HistogramEventTypes	Array of events to be considered by the model.	["print-activity"]
Disabled	Determines whether the model should be enabled. If TRUE no data is collected.	FALSE

1.1.1. Model Example

Refer to this example to learn more about model attributes.

Here is an example model containing many of the attributes described in the model attributes table.

```
PR-UP {
    ModelTemplate = "Printers for user"
    Description = "Models the printers used by this user."
    Category = "Print Activity"
    IconName = ""
    ScopeType = "USER"
    Scope = """user"""
    Feature = """printer_name"""
    FeatureName = "printer"
    FeatureType = "printer"
    TrainIf = """count(printer_name,'print-activity')=1"""
    ModelType = "CATEGORICAL"
    AgingWindow = ""
    CutOff = "5"
    MaxNumberOfBins = "20000"
    Alpha = "1"
```

```
  ConvergenceFilter = "confidence_factor>=0.8"
  HistogramEventTypes = [
    "print-activity"
  ]
  Disabled = "FALSE"
 }
```

1.2. RULE ATTRIBUTES

Refer to this table to learn more about rule attributes.

Attribute	Definition	Example
Rule Id	ID of the rule, which must be unique. This value can be used when searching for a rule in Threat Hunter. `rule_id_name { all the \n below \n fields }`	PR-UP-F
RuleName	Free text describing the rule. This text appears in the UI when a rule triggers several times in a session and is aggregated. It can also be used to identify a rule in Threat Hunter.	First print activity from printer for user
RuleDescription	Text providing more details about the rule. This appears in the UI when the rule details are expanded.	This is the first time for this user to print from this printer. This can be significant because printing can be a way to exfiltrate data from the organization
ReasonTemplate	Text that appears in the UI that explains the rule. `{default|featureValue|histogram}` is a placeholder that is replaced with event specific values when rendered in the UI. The first part, default, indicates there is no special treatment of the value. Other options are asset, location.country, location.zone, time.day_of_week, time.time_of_week, user, or user.group. `featureValue` is replaced with the feature value, printer_name in this case. `scopeValue` or event.<field_name> can also be used. The field has to be persisted in Mongo for the value to show up correctly. `|histogram` is an optional part which makes the value clickable and will link to the model instance.	First print activity from printer for user: {default|featureValue|histogram}
AggregateReasonTemplate	Rules that trigger multiple times in a session will be aggregated when reviewing the session in the user page. The header of the aggregated rules is the `RuleName` field. When expanding the aggregated rule this is the text that will show up. The event specific placeholder is the same as in `ReasonTemplate`.	First print activity from printer for user: {default|featureValue|histogram}
RuleType	Session or sequence the rule should trigger in. Possible values are account-lockout, asset, database, endpoint, file, session, or web.	session
RuleCategory	Free text field describing use case/classification. Rules are grouped under this value in the rule editor UI.	Data Loss Prevention

Attribute	Definition	Example
ClassifyIf	Expression indicating how often the rule should trigger. The expression in this example means once per printer name. The syntax and logic is the same as for the model `TrainIf` attribute. **NOTE** For fact based rules this value, and all conditions placed in the `RuleExpression` attribute, has to be TRUE.	count(printer_name, 'print-activity')=1
RuleEventTypes	Array indicating which event could trigger the rule.	["print-activity"]
Disabled	Whether the rule is enabled or not.	FALSE
Model	The model used by the rule. For fact based rules this should be FACT.	PR-UP
FactFeatureName	For fact based rules only, this value will be shown when using `featureValue` in the `ReasonTemplate` and `AggregateReasonTemplate` fields.	printer_name
Score	How much the rule should be scored. Starting in Advanced Analytics I46, this can be an expression, for example multiply(field1,field2). Negative values are also allowed to reduce session risk. This score will be adjusted based on the data if Histogram shaping and Bayesian scoring are enabled.	10.0
RuleLabels	Rule tagging. Currently used for Mitre Att&ck feature.	mitre = ["T1052"]
PercentileThreshold	Percentile below which values are considered anomalous. Will affect the value of the `percentile_threshold_count` (p-value) expression, such as in the expression `num_observations<percentile_threshold_count`.	0.1
RuleExpression	Expression that defines when the rule should trigger. In model based rules `num_observations` indicates how many times the current value (feature) was observed. 0 means never observed before (first). Other model based expressions are: • **num_observations** – the number of times this feature appears in the model. • **probability** – the number of times the current value exists in the model divided by the total data points in the model. • **total_events** – the number of data points in the model. • **num_bins**– the number of bins in the model. • **confidence_factor** – the result of the calculation ((N-C)/N)^a in which N=total data points in the model, C=number of bins, and a=alpha. • **ConfidenceFactorAboveOrEqual()** – returns true if `confidence_factor` is above or equal 0.8. The 0.8 threshold is defined in the `GlobalConfidenceFactor` parameter in the configuration file. `ConfidenceFactorAboveOrEqual(n)` can also be used to specify a different threshold.	num_observations=0 && ConfidenceFactorAboveOrEqual()

Attribute	Definition	Example
DependencyExpression	Whether the rule should trigger only if a different rule has triggered (or not) on the same event. Other rules are referenced by Id and can be used in boolean operations, for example (R1 || R2) && !R3.	NA
ScoreTarget (optional)	For asset based rules, with events with both a dest and src, send the points to only the one specified in `scoreTarget`.	scoreTarget = src_host

1.2.1. Rule Example

The following example contains many of the attributes described in the rule attributes table.

```
PR-UP-F {
    RuleName = "First print activity from printer for user"
    RuleDescription = "This is the first time for this user to print from this
printer"
    ReasonTemplate = "First print activity from printer {default|featureValue|
histogram} for user"
    AggregateReasonTemplate = "First print activity from printer for user:
{default|featureValue|histogram}"
    RuleType = "session"    RuleCategory = "Data Loss Prevention"
    ClassifyIf = """count(printer_name, 'print-activity')=1"""
    RuleEventTypes = [
      "print-activity"
    ]
    Disabled = "FALSE"
    Model = "PR-UP"
    FactFeatureName = "printer_name"
    Score = "10.0"
    RuleLabels {
       mitre = ["T1052"]
    }
    PercentileThreshold = "0.1"
    RuleExpression = """num_observations=0 && ConfidenceFactorAboveOrEqual()"""
    DependencyExpression = "NA"
  }
```

2. Event Types and Required Fields

Refer to this appendix to learn more about the required fields in the event for every event type.

This appendix defines the required fields that should be present in the event for every event type.

NOTE

Events can still be created if the required fields are not present. However, the event will not apply from a rule scoring and modeling standpoint.

EventName	Description	Required Fields	Optional Fields
process-network-failed	An endpoint process was blocked from accessing a network.	• host • time • dest_host/dest_ip • src_host/src_ip • process_name	• direction • src_port • process • bytes • domain • user • event_code • event_name • process_directory • dest_port
network-connection-failed	A network connection failure occurred.	• host • time • src_host/src_ip • dest_host/dest_ip • action • src-port • dest_port	• direction • src_interface • protocol • event_name • src_translated_port • bytes_in • dest_translated_ip • rule • src_mac • bytes • dest_mac • user • event_code • bytes_out • dest_interface • outcome • src_translated_ip • dest_translated_port
database-login	A user logged into the database.	• host • time • database_name • db_user • user	• src_host/src_ip • domain • protocol • dest_host/dest_ip • service_name • app • process_name • process • event_code • server_group • event_name

EventName	Description	Required Fields	Optional Fields
database-delete	One or more records were deleted from the database.	• host • time • database_name • db_user • user	• src_host/scr_ip • process • database_schema • dest_host/dest_ip • app • process_name • domain • db_operation • table_name • event_code • database_object • server_group • event_name
privileged-object-access	A user obtained special privileges to access a privileged object. **NOTE** This is tied to Windows events 4674 or 578.	• host • time • user • dest_host/dest_ip • privileges • object	• ownership_privilege • src_host/src_ip • domain • object_type • event_code • process • environment_privilege • process_name • object_server • logon_id • tcb_privilege • debug_privilege • event_name • process_directory
account-creation	A user created a new account. **NOTE** This is tied to Windows events 4720 or 624.	• host • time • dest_host/dest_ip • account_name • user	• src_host/src_ip • domain • event_name • logon_id • event_code • account_domain

EventName	Description	Required Fields	Optional Fields
dns-query	An asset queried for a domain in the DNS server.	• host • time • src_host/src_ip • query	• src_port • dest_host/dest_ip • query_type • bytes • event_name • query_flags • user • event_code • src_mac • dest_port
vpn-logout	A user logged off remote access VPN.	• host • time • user	• src_host/src_ip • domain • session_duration • realm • dest_host/dest_ip • bytes_out • session_id • event_name • event_code • bytes_in • os • src_translated_ip
dlp-email-alert-out	Outgoing email activity reported by an email monitoring tool.	• host • time • recipient • sender	• src_host/src_ip • direction • external_domain • attachments • recipients • dest_host/dest_ip • bytes • num_recipients • return_path • event_name • user • event_code • external_address • outcome • message_id • user_email • subject

EventName	Description	Required Fields	Optional Fields
service-logon	A non-interactive service logon occurred. **NOTE** This is tied to Windows events 4624 and 528 with logon type 5.	• host • time • dest_host/dest_ip • user	• src_host/src_ip • domain • event_code • auth_package • process • process_name • logon_id • logon_type • event_name
dlp-alert	An alert was reported by a DLP product running on the endpoints.	• host • time • src_host/src_ip • alert_name	• alert_severity • alert_type • target • dest_host/dest_ip • file_name • domain • event_name • alert_id • user • event_code • outcome • additional_info
file-write	A file was created, edited, or moved.	• host • time • dest_host/dest_ip • file_name • user	• src_host/src_ip • domain • file_type • app • process_name • bytes • src_file_name • accesses • file_path • process • event_code • activity • file_ext • src_file_dir • event_name • file_parent

EventName	Description	Required Fields	Optional Fields
account-switch	A user switched their account to impersonate another account. **NOTE** This is tied to Windows events 4648 and 552. Also tied to Unix SUDO logs.	• host • time • dest_host/dest_ip • account • user	• src_host/src_ip • domain • safe_value • process • account_logon_guid • process_name • dest_service • user_uid • user_logon_guid • user_sid • logon_id • event_code • account_domain • event_name • process_directory
authentication-failed	An authentication attempt performed either from a public IP address or from an internal network address failed.	• host • time • dest_host/dest_ip • user • failure_reason	• src_host/src_ip • domain • app • additional_info • event_name • user_agent • event_code • outcome • os • auth_method • browser

EventName	Description	Required Fields	Optional Fields
file-download	A file was downloaded.	• host • time • dest_host/dest_ip • file_name • user	• src_host/src_ip • domain • file_type • app • process_name • bytes • src_file_name • accesses • file_path • process • event_code • activity • file_ext • src_file_dir • event_name • file_parent
app-login	A user logged into an application.	• host • time • app • user	• src_host/src_ip • protocol • dest_host/dest_ip • event_name • user_agent • event_code • os • user_email

EventName	Description	Required Fields	Optional Fields
usb-write	A user copied files from their machine to a USB flash drive.	• host • time • dest_host/dest_ip • file_name • user	• src_host/src_ip • src_file_name • activity_details • event_code • process • process_name • bytes • domain • src_file_ext • device_type • activity • file_ext • src_file_dir • event_name • file_path • process_directory • device_id
database-alert	Abnormal activity in the database was detected either by Exabeam or by a third-party monitoring tool.	• host • time • alert_name • db_user • user • database_name	• domain • alert_type • event_code • dest_host/dest_ip • service_name • app • process_name • process • alert_id • database_object • server_group • event_name • additional_info • src_host/src_ip • alert_severity • malware_url • db_operation • table_name • response_size

EventName	Description	Required Fields	Optional Fields
print-activity	A user printed files, data, or some other form of content.	• host • time • user • dest_host/dest_ip • printer_name • object	• src_host/src_ip • domain • num_pages • bytes • event_name • event_code • activity • outcome
workstation-locked	A user locked their workstation. **NOTE** This is tied to Windows event 4800.	• host • time • dest_host/dest_ip • user	• src_host/src_ip • domain • event_code • event_name
failed-physical-access	A user swiped their physical badge to open a door, gate, or other entrance but were denied access.	• host • time • outcome • badge_id • location_door	• location_city • location_building • first_name • last_name • src_host/src_ip • dest_host/dest_ip • event_name • employee_id • user • event_code
vpn-connection	A user used VPN to connect to a network.	• host • time • src_host/src_ip • src_port • dest_host/dest_ip • dest_port	• event_name • event_code • bytes_in • dest_translated_ip • session_id • duration • user • access_group • bytes_out • action • src_translated_ip

EventName	Description	Required Fields	Optional Fields
app-activity	A user's activity within a specific application.	• host • time • user • activity • app • object	• src_host/src_ip • resource • dest_host/dest_ip • event_name • result • event_code • additional_info • user_agent
usb-insert	A USB flash drive was connected to the network.	• host • time • dest_host/dest_ip • user • device_id	• src_host/src_ip • domain • activity_details • event_code • process • process_name • bytes • device_type • event_name
dlp-email-alert-in-failed	An inbound email activity failure. For example, if there is an email server error.	• host • time • recipient • sender	• src_host/src_ip • direction • external_domain • attachments • recipients • dest_host/dest_ip • bytes • num_recipients • return_path • event_name • user • event_code • external_address • outcome • message_id • user_email • subject

EventName	Description	Required Fields	Optional Fields
account-unlocked	An administrator unlocked a user's account.	• host • time • target_user • user	• src_host/src_ip • domain • user_sid • target_domain • dest_host/dest_ip • logon_id • event_code • outcome • event_name
process-alert	A user has executed a process that triggered an organization's configured endpoint process alert.	• host • time • dest_host/dest_ip • alert_name • process_name	• src_host/src_ip • domain • alert_type • process • command_line • alert_severity • parent_process • alert_id • user • event_code • process_directory • event_name • md5
privileged-access	A user obtained special privileges. For example, if a regular user who does not have administrator privileges attempts to elevate their own privileges to have administrator privileges. **NOTE** This is tied to events indicating privileged access or service, such as Windows events 4672, 4673, 576, and 577.	• host • time • dest_host/dest_ip • privileges • user	• ownership_privilege • src_host/src_ip • domain • event_code • process • environment_privilege • process_name • object_server • logon_id • tcb_privilege • debug_privilege • event_name • process_directory

EventName	Description	Required Fields	Optional Fields
dlp-email-alert-out-failed	An outbound email activity failure occurred. For example, if the recipient email address is wrong or if there is an email server error.	• host • time • recipient • sender	• src_host/src_ip • direction • external_domain • attachments • recipients • dest_host/dest_ip • bytes • num_recipients • return_path • event_name • user • event_code • external_address • outcome • message_id • user_email • subject
dns-response	An asset received a response from a DNS server.	• host • time • dest_host/dest_ip • query • dns_response_code	• src_host/src_ip • src_port • query_type • bytes • event_name • query_id • user • event_code • response_flags • response • dest_port
database-failed-login	A user attempted and failed to log in to a database.	• host • time • reason • db_user • user • database_name	• src_host/src_ip • domain • dest_host/dest_ip • service_name • app • process_name • process • event_code • server_group • outcome • event_name

EventName	Description	Required Fields	Optional Fields
process-created	A user has executed an endpoint process on a host.	• host • time • dest_host/dest_ip • process_name	• src_host/src_ip • domain • process • pid • command_line • parent_process • logon_id • user • event_code • path • event_name • process_directory • md5
failed-vpn-login	A remote access VPN login attempt performed either from a public IP address or from an internal network address failed.	• host • time • src_host/src_ip • user	• domain • realm • dest_host/dest_ip • failure_reason • event_name • event_code
nac-failed-logon	A logon attempted to a NAC failed.	• host • time • dest_host/dest_ip • domain • user	• src_host/src_ip • network • event_code • auth_server • event_name

EventName	Description	Required Fields	Optional Fields
web-activity-denied	A user was blocked by a restricting policy while attempted to access a web resource via a proxy or other web monitoring gateway.	• host • time • user • action • method • web_domain	• protocol • dest_host/dest_ip • bytes_out • uri_path • proxy_action • mime • categories • dest_port • category • src_host/src_ip • top_domain • src_port • referrer • result_code • failure_reason • src_ip • event_name • user_agent • event_code • bytes_in • os • full_url • uri_query
failed-app-login	A user failed to log in to an application.	• host • time • app • user • failure_reason	• src_host/src_ip • dest_host/dest_ip • event_name • user_agent • event_code • outcome • os • browser

EventName	Description	Required Fields	Optional Fields
account-password-change	A user changed their account password. **NOTE** This is tied to Windows events 4723 or 627.	• host • time • target_user • user	• src_host/src_ip • domain • user_sid • target_domain • dest_host/dest_ip • logon_id • event_code • outcome • event_name
account-deleted	A user deleted an account. **NOTE** This is tied to Windows events 4726 or 630.	• host • time • target_user • user	• src_host/src_ip • domain • user_sid • target_domain • dest_host/dest_ip • logon_id • event_code • event_name
file-read	A user opened or downloaded a file.	• host • time • dest_host/dest_ip • file_name • user	• src_host/src_ip • domain • file_type • app • process_name • bytes • src_file_name • accesses • file_path • process • event_code • activity • file_ext • src_file_dir • event_name • file_parent

EventName	Description	Required Fields	Optional Fields
usb-read	SB read activity was detected.	• host • time • user • dest_host/dest_ip • file_name • device_id	• src_host/src_ip • domain • activity_details • event_code • process • process_name • bytes • device_type • activity • file_ext • event_name • file_path • process_directory
nac-logon	A user was granted network access.	• host • time • dest_host/dest_ip • user	• auth_type • src_host/src_ip • domain • network • event_name • event_code • auth_server • src_mac
share-access-denied	This user has been denied access to a Windows network share.	• host • time • user • share_name • dest_host/dest_ip • outcome	• src_host/src_ip • domain • share_path • file_type • file_name • accesses • logon_id • event_code • event_name

EventName	Description	Required Fields	Optional Fields
file-alert	A file integrity product (such as Tripwire) reported a change made to critical and/or system file.	• host • time • dest_host/dest_ip • file_name • alert_name	• src_host/src_ip • domain • alert_type • process • process_name • alert_severity • accesses • alert_id • user • event_code • file_ext • file_parent • event_name • file_path
audit-log-clear	An audit log was deleted from the system. **NOTE** This is tied to Windows events indicating audit log clearance, such as Windows 1102 and 517.	• host • time • dest_host/dest_ip • user	• src_host/src_ip • domain • logon_id • event_code • event_name
local-logon	A local logon occurred. **NOTE** This is tied to Windows events 4624 or 528 events with logon type 2 or 7. Also tied to Windows events with logon type 11 and a process name indicating a local interactive logon. And tied to Linux local logon events.	• host • time • dest_host/dest_ip • user	• src_host/src_ip • domain • event_code • auth_package • process • process_name • logon_id • logon_type • event_name
ntlm-logon	An interactive logon using NTLM authentication occurred. **NOTE** This is tied to Microsoft NTLM events that indicate an interactive logon by user, such as Windows events 4776 or 680. For more precise readings on the nature of the logon, consider collecting Windows 4624 events from the asset.	• host • time • domain • user • dest_host/dest_ip • result_code	• src_host/src_ip • event_name • event_code

EventName	Description	Required Fields	Optional Fields
winsession-disconnect	A user disconnected from an existing Terminal Services session. **NOTE** This is tied to Windows event 4779.	• host • time • dest_host/dest_ip • domain • user	• src_host/src_ip • logon_id • event_code • event_name
service-created	A service was installed on the system. **NOTE** This is tied to service creation events, such as Windows 4697.	• host • time • dest_host • service_name • user	• src_host/src_ip • process • process_name • dest_host/dest_ip • user_sid • logon_id • event_code • service_type • account_domain • event_name • account_name • process_directory
batch-logon	A non-interactive batch logon occurred. **NOTE** This is tied to Windows events 4624, and 528 with logon type 4.	• host • time • dest_host/dest_ip • user	• src_host/src_ip • domain • event_code • auth_package • process • process_name • logon_id • logon_type • event_name
computer-logon	A non-interactive computer logon occurred.	• host • time • dest_host/dest_ip • user	• src_host/src_ip • domain • event_code • event_name

EventName	Description	Required Fields	Optional Fields
ds-access	User accessed an active directory object.	• host • time • domain • object • user	• old_attribute • src_host/src_ip • object_dn • dest_host/dest_ip • attribute • new_attribute • event_name • logon_id • event_code • object_ou • object_class • activity_type
process-created-failed	A user failed to execute an endpoint process on a host.	• host • time • dest_host/dest_ip • process_name	• src_host/src_ip • domain • process • pid • command_line • parent_process • logon_id • user • event_code • path • outcome • event_name • process_directory • md5
kerberos-logon	An interactive logon using Kerberos occurred. **NOTE** This is tied to Windows events 4768 or 672. For more precise readings on the nature of the logon, consider collecting Windows events 4624 from the asset.	• host • time • domain • user • dest_host/dest_ip • result_code	• src_host/src_ip • user_sid • service_name • ticket_encryption_type • event_code • event_name • ticket_options

EventName	Description	Required Fields	Optional Fields
failed-usb-activity	USB activity failed. For example, an administrator sets a policy to deny USB activity on machines connected to the company network. Then, a user attempts to copy files to a USB flash drive and is denied by the policy. The activity would be logged as failed-USB-activity.	• host • time • dest_host/dest_ip • user • device_id	• src_host/src_ip • domain • activity_details • event_code • process • process_name • bytes • device_type • activity • event_name
security-alert	An alert was reported by a third-party security product, such as FireEye, Palo Alto Networks, or other antivirus software running on the endpoints.	• host • time • src_host/src_ip • alert_name	• alert_type • alert_sevirity • dest_host/dest_ip • file_name • process_name • malware_url • process • alert_id • user • event_code • event_name • additional_info
app-activity-failed	A user successfully logged in to an app but failed to perform an action in the app.	• host • time • user • activity • app • outcome	• src_host/src_ip • resource • dest_host/dest_ip • object • event_name • user_agent • event_code • additional_info • result
vpn-login	Remote access VPN login attempt either from a public IP address or from an internal network address was successful.	• host • time • src_host/src_ip • user	• domain • realm • dest_host/dest_ip • os • session_id • event_name • event_code • src_translated_ip

EventName	Description	Required Fields	Optional Fields
physical-access	A user successfully opened a door, gate, or other entrance using their badge.	• host • time • badge_id • location_door	• location_city • location_building • first_name • last_name • src_host/src_ip • dest_host/dest_ip • event_name • employee_id • user • event_code • outcome
file-upload	A file was uploaded to the web.	• host • time • dest_host/dest_ip • file_name • user	• src_host/src_ip • domain • file_type • app • process_name • bytes • src_file_name • accesses • file_path • process • event_code • activity • file_ext • src_file_dir • event_name • file_parent
config-change	A user made a configuration change.	• host • time • user • activity • dest_host/dest_ip • object	• src_host/src_ip • event_code • outcome • event_name

EventName	Description	Required Fields	Optional Fields
database-update	A user issued a database query to update one or more database records.	• host • time • database_name • db_user • user	• src_host/src_ip • process • database_schema • dest_host/dest_ip • app • process_name • domain • db_operation • table_name • event_code • database_object • server_group • event_name
account-password-change-failed	A user attempted to change their account password but failed. **NOTE** This is tied to Windows events 4723 or 627.	• host • time • target_user • user	• target_user_sid • domain • src_host/src_ip • user_sid • target_domain • dest_host/dest_ip • logon_id • event_code • outcome • event_name
database-activity-failed	A database query was issued and then failed.	• host • time • db_user • user • db_operation • database_name • outcome	• src_host/src_ip • domain • resource • dest_host/dest_ip • app • process_name • process • event_code • server_group • database_schema • event_name

EventName	Description	Required Fields	Optional Fields
network-alert	Suspicious activity in the network was detected and reported by a network security product, such as an IDS or IPS.	• host • time • dest_host/dest_ip • src_host/src_ip • alert_name	• domain • alert_type • src_port • alert_severity • event_name • user • event_code • protocol • additional_info • dest_port
web-activity-allowed	A user has accessed a web resources via a proxy or some other web monitoring gateway.	• host • time • user • action • method • web_domain	• protocol • dest_host/dest_ip • bytes_out • uri_path • proxy_action • mime • categories • dest_port • category • src_host/src_ip • top_domain • src_port • referrer • result_code • src_ip • event_name • user_agent • event_code • bytes_in • os • full_url • uri_query

EventName	Description	Required Fields	Optional Fields
remote-logon	A remote, interactive logon occurred. **NOTE** This is tied to Windows events 4624 with logon type 10 or 11. Also tied to Unix SSH login events.	• host • time • dest_host/dest_ip • src_host/src_ip • user	• domain • event_code • auth_package • process • service_name • process_name • logon_id • logon_type • event_name • ticket_options
failed-logon	A user failed a logon attempt.	• host • time • dest_host/dest_ip • user	• src_host/src_ip • domain • event_code • process • process_name • failure_reason • user_sid • logon_type • event_name • result_code
account-enabled	An account was enabled by a user.	• host • time • dest_host/dest_ip • target_user • user	• src_host/src_ip • domain • target_domain • event_name • logon_id • event_code
audit-policy-change	An audit policy was changed. **NOTE** This is tied to Windows events 4719 and 612.	• host • time • policy • domain • user • dest_host/dest_ip	• src_host/src_ip • event_code • subcategory • event_name • logon_id • audit_category
workstation-unlocked	A user unlocked their workstation. **NOTE** This is tied to Windows event 4801.	• host • time • dest_host/dest_ip • user	• src_host/src_ip • domain • event_code • event_name

EventName	Description	Required Fields	Optional Fields
database-query	A user queried a database.	• host • time • db_user • user • database_name • db_query	• src_host/src_ip • process • database_schema • event_code • dest_host/dest_ip • app • process_name • domain • db_operation • table_name • response_size • database_object • server_group • event_name
member-added	A user has been added to a domain group membership.	• host • time • user • account_id • group_name	• src_host/src_ip • account_ou • dest_host/dest_ip • group_domain • domain • event_name • logon_id • event_code • account_dn • group_type
remote-access	A remote, non-interactive logon occurred. **NOTE** This is tied to Windows events 4769, or 4624 with logon type 3 or 8.	• host • time • src_host/src_ip • user • service_name	• domain • event_code • auth_package • dest_host/dest_ip • process_name • ticket_encryption_type • process • logon_id • logon_type • event_name • ticket_options

EventName	Description	Required Fields	Optional Fields
account-disabled	An administrator disabled a user's account.	• host • time • target_user • user	• src_host/src_ip • domain • user_sid • target_domain • dest_host/dest_ip • logon_id • event_code • event_name
network-connection-successful	A network connection attempt was successful.	• host • time • src_host/src_ip • dest_host/dest_ip • src-port • dest_port	• direction • src_interface • protocol • event_name • src_translated_port • bytes_in • bytes • bytes_out • src_mac • dest_mac • rule • event_code • dest_translated_ip • action • dest_interface • outcome • src_translated_ip • dest_translated_port • user
authentication-successful	An authentication attempt performed either from a public IP address or from an internal network address was successful.	• host • time • dest_host/dest_ip • user	• src_host/src_ip • domain • app • event_name • user_agent • event_code • outcome • os • auth_method • browser

EventName	Description	Required Fields	Optional Fields
netflow-connection	A new NetFlow connection was detected.	• host • time • src_host/src_ip • dest_host/dest_ip • src-port • dest_port	• direction • src_interface • protocol • bytes_out • packets • time_end • bytes • end_reason • user • event_code • bytes_in • time_start • dest_interface • outcome • event_name
share-access	This user has accessed a Windows network share.	• host • time • dest_host/dest_ip • user • share_name	• src_host/src_ip • domain • share_path • file_type • file_name • accesses • logon_id • event_code • outcome • event_name
account-password-reset	An administrator reset a user's password. **NOTE** This is tied to Windows events 4724 or 628.	• host • time • target_user • user	• target_user_sid • domain • src_host/src_ip • user_sid • target_domain • dest_host/dest_ip • logon_id • event_code • outcome • event_name

EventName	Description	Required Fields	Optional Fields
file-permission-change	A user has changed the permissions for a file and/or folder.	• host • time • user • dest_host/dest_ip • file_name • accesses	• src_host/src_ip • domain • file_type • app • process_name • bytes • file_path • process • event_code • activity • file_ext • event_name • file_parent
account-lockout	An account has been locked.	• host • time • dest_host/dest_ip • domain • user	• src_host/src_ip • caller_user • caller_domain • logon_id • event_code • event_name • auth_method
database-access	A user accessed a database.	• host • time • db_user • user • db_operation • database_name	• src_host/src_ip • domain • database_schema • dest_host/dest_ip • sql_count • app • process_name • session_id • process • table_name • event_code • service_name • database_object • server_group • event_name • additional_info

EventName	Description	Required Fields	Optional Fields
failed-ds-access	An access attempt to an active directory object failed.	• host • time • domain • object • user	• src_host/src_ip • object_dn • dest_host/dest_ip • attribute • failure_reason • event_name • event_code • object_ou • outcome • object_class • activity_type
member-removed	A user has been removed from a domain group membership.	• host • time • user • account_id • group_name	• src_host/src_ip • account_ou • dest_host/dest_ip • group_domain • domain • event_name • logon_id • event_code • account_dn • group_type
task-created	A user created a new scheduled task. **NOTE** Tied to Windows event 4698.	• host • time • dest_host/dest_ip • user • task_name	• src_host/src_ip • domain • description • process • process_name • run_level • event_code • account_domain • event_name • account_name • process_directory

EventName	Description	Required Fields	Optional Fields
process-network	A process executing on the endpoint tried to access the network.	• host • time • dest_host/dest_ip • src_host/src_ip • process_name	• direction • src_port • process • bytes • domain • user • event_code • event_name • process_directory • dest_port
file-delete	A user deleted a file.	• host • time • dest_host/dest_ip • file_name • user	• src_host/src_ip • domain • file_type • app • process_name • bytes • accesses • file_path • process • event_code • activity • file_ext • event_name • file_parent