Forcepoint Behavioral Analytics Product Configuration Manual

Product Configuration | Forcepoint Behavioral Analytics | v3.3.x, v3.4.x | 17-Feb-2021

Configuration Overview

This Configuration Manual outlines precisely defined standards for client data that Forcepoint Behavioral Analytics ingests, as well as the flexible processes for moving, enriching, and displaying this data. Interactions with these configurable components can exist as command line prompts, HTTP requests, or changing file attributes.

Single Sign On (SSO)

Forcepoint Behavioral Analytics uses SAML 2.0 to manage SSO (supported over PKI) user authentication for Forcepoint Behavioral Analytics application access.

Data Ingest and Processing

The Streaming Ingest platform transforms source data into Forcepoint Behavioral Analytics Events and then scores and persists those Events, making them available in the Forcepoint Behavioral Analytics user interface. The platform exposes a REST API for receiving and real-time processing of source data.

In addition to the REST API, Forcepoint Behavioral Analytics includes Apache NiFi, a highly scalable, resilient, and extensible platform, for publishing data to the REST API from various data sources.

Data Storage and Query

Forcepoint Behavioral Analytics uses Elasticsearch and PostgreSQL to house application data, and uses both internal and external APIs for inserting and querying this data. Elasticsearch stores Event data, while PostgreSQL contains user information, such as account details and permissions, along with analytics caching and UI data (Saved Searches, Entity Filters, notifications). A streaming ingest pipeline allows for continuous processing of Events, (as opposed to batch processing) culminating in a Logstash service that inserts Event data into Elasticsearch. The User

Interface (UI) interacts with the Master Data Service (MDS) to retrieve and display these Events as needed.

User Interface

The Forcepoint Behavioral Analytics User Interface (UI) has several key configuration settings accessible through a file called **AppConfig**. This file contains several JSON objects that control Feature Scoring, data display, and data visualization specifics.

Data Display - Event Viewer

The Event Viewer is the User Interface component that shows details of ingested events.

By default, the Event Viewer shows all event roles and attributes, with attribute names and roles as stored with the event. Furthermore, the roles appear in alphabetical order based on role names.

Event Viewer - Event Role Configuration

It is possible to change the display names and display order of roles, as well as to make certain roles hidden.

Here are some examples of the process to make these changes:

1. Defining the order of roles:

```
  curl -H "Content-Type:application/json" -XPUT http://
localhost:8080/reference/config/mds -d
'{"entity_roles_config":{"sender":{"order":1}}}'
```

```
  curl -H "Content-Type:application/json" -XPUT http://
localhost:8080/reference/config/mds -d
'{"entity_roles_config":{"recipient":{"order":2}}}'
```

2. Defining order and overriding the displayed name:

```
  curl -H "Content-Type:application/json" -XPUT http://
localhost:8080/reference/config/mds -d
'{"entity_roles_config":{"recipientCc":{"order":3,
"displayName":"CC"}}}'
```

3. Defining just a display name override:

```
  curl -H "Content-Type:application/json" -XPUT http://
localhost:8080/reference/config/mds -d
'{"entity_roles_config":{"recipientBcc":{"displayName":"B
CC"}}}'
```

4. Hiding a role:

```
  curl -H "Content-Type:application/json" -XPUT http://
localhost:8080/reference/config/mds -d
'{"entity_roles_config":{"recipient":{"hidden":true}}}'
```

5. Reset any changes and go back to defaults:

```
curl -H "Content-Type:application/json" -XPUT http://
localhost:8080/reference/config/mds -d
'{"entity_roles_config":{"sender":null}}'
```

Event Viewer - Event Attribute Configuration

At this time, "hidden" is the only supported option for event attributes. Case and spelling of attribute names matter and must be an exact match. Here are some examples:

1. Hiding an attribute in the Event Viewer

   ```
   curl -XPUT http://localhost:8080/reference/config/mds -d
   '{"event_attributes_config":{"Duration":{"hidden":true}}}
   '
   ```

2. Hiding several attributes in the Event Viewer using one object

   ```
   curl -XPUT http://localhost:8080/reference/config/mds -d
   '{"event_attributes_config":{"Duration":{"hidden":true},
   "Total Bytes":{"hidden":true}}}'
   ```

3. Reset any changes and go back to defaults:

   ```
   curl -XPUT http://localhost:8080/reference/config/mds -d
   '{"event_attributes_config":null}'
   ```

Data Management

There are several systems in place to help move and organize Forcepoint Behavioral Analytics data.

- Timeslicing organizes large amounts of Event data into manageable, chronologically separated chunks.
- The Minigator migrates Forcepoint Behavioral Analytics Event, Entity, and metadata to new application versions during upgrades.
- Knex is a third party tool responsible for managing UI PostgreSQL migrations and schema changes.

Application Monitoring

Forcepoint Behavioral Analytics has several systems in place to monitor operations and performance, as well as log files to track backend activity. This helps to understand functional details, efficiently address bugs, and identify potential need for scaling.

Appendix (Data Format)

Forcepoint Behavioral Analytics stores each Event, Entity, and metadata object (such as Feature, Model, and Lexicon configurations) in a standardized JSON format.

Single Sign On (SSO)

Single Sign-On (SSO) is a session/user authentication process that permits a user to enter one name and password in order to access multiple applications in a company's IT infrastructure.

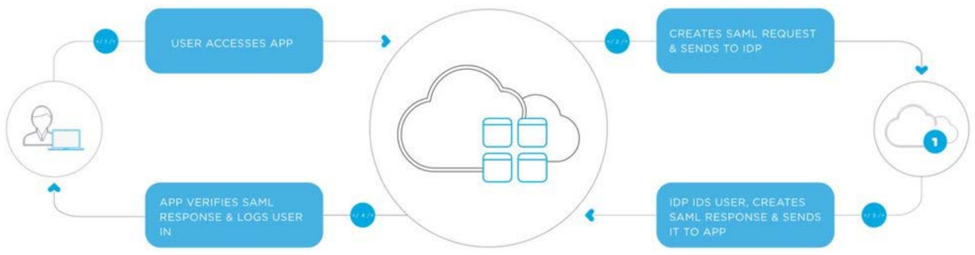
Security Assertion Markup Language (SAML) is an XML standard that allows secure web domains to exchange user authentication and authorization data. Using SAML, an online service provider can contact a separate online identity provider to authenticate users who are trying to access secure content. SAML supports multiple SSO providers (e.g., OneLogin, Active Directory) so that users can access the app (authenticate) with a single click.

Service Provider (SP) is an application that will use a separate identity provider for authentication. In our case, the Forcepoint Behavioral Analytics UI is the Service Provider. In the diagram below, Service Provider is referred to as "app".

Identity Provider (IdP) is the SSO solution that manages identities. In our case, OneLogin is the Identity Provider. In the diagram below, Identity Provider is referred to as "IDP".

User is a given individual, particularly his/her browser session.

How SAML Works


USER ACCESSES APP
CREATES SAML REQUEST & SENDS TO IDP
IDP IDS USER, CREATES SAML RESPONSE & SENDS IT TO APP
APP VERIFIES SAML RESPONSE & LOGS USER IN
1

1. The user accesses the remote application using a link and the application loads.
2. App identifies the user's origin (by application subdomain, IP address, or similar) and redirects the user back to the Identity Provider, asking for authentication. This is the authentication request.
3. The user either has an existing active browser session with the Identity Provider or establishes one by logging into the Identity Provider.
4. The Identity Provider builds the authentication response in the form of an XML-document containing the user's username or email address, signs it using an X.509 certificate, and posts this information to the Service Provider.
5. The Service Provider, which already knows the Identity Provider and has a certificate fingerprint, retrieves the authentication response and validates this using the certificate fingerprint.
6. The identity of the user is established and the user is provided with App access.

SAML As Seen In Forcepoint Behavioral Analytics

Configuration/setup

SSO_TYPE: "saml"

saml_config with entryPoint

- Based on these values, we use passport-saml strategy that corresponds to correct Identity Provider

User Accesses App

1. Navigate to Forcepoint Behavioral Analytics.
2. If no session stored, redirect to /login.
3. Click **Get started**.

Creates SAML Request and Sends to IDP

1. Request to /session/new in node. passport-saml strategy has been configured.
2. Browser request translates into an encoded SAML message.
3. That message becomes an authorization request. It's a request to a URL that is a combination of:
   a. entryPoint, like https://redowl.onelogin.com/trust/saml2/http-post/sso/533049, as configured in saml_config.coffee, and
   b. query param of SAMLRequest={message from above}
4. The node server redirects to that URL, which takes us to OneLogin.

IdP IDs User, Creates SAML Response, and Sends to App

1. If the user is already logged into OneLogin, OneLogin (as Identity Provider) recognizes this and sends SAML response with the user's name and email back to Forcepoint Behavioral Analytics (as Service Provider).
2. If the user is not logged into OneLogin, OneLogin asks for the user's credentials (email/password/OTP). If the user logs in successfully, OneLogin sends SAML response with the user's name/email back to Forcepoint Behavioral Analytics (as Service Provider).

App Verifies SAML Response and Logs User In

1. Forcepoint Behavioral Analytics receives the SAML response from OneLogin.
2. If Forcepoint Behavioral Analytics sees an email in the SAML message, Forcepoint Behavioral Analytics checks for that email in Postgres. If found, the user is serialized into a login session and logged into Forcepoint Behavioral Analytics. Authentication is complete.

3. If Forcepoint Behavioral Analytics doesn't find an email from the SAML message in Postgres, the user is not logged in and must try again. This could be the result of:
    a. Bad SAML response without email, or
    b. The user's OneLogin email address is not matching email address in Postgres

Configuration and Operational Details

Given a browser request (User clicks "get started" and POST to /session/new), the passport-saml strategy will:

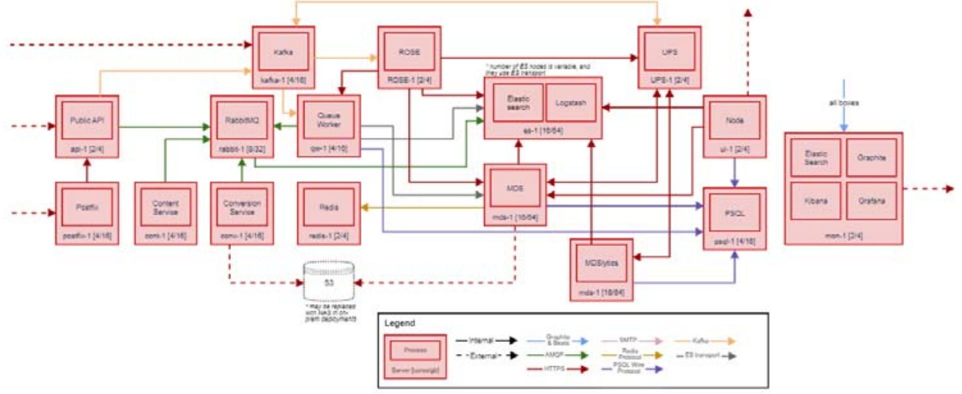
1. Generate authorization request which includes:
    a. issuer
    b. callback route (in our case, /api/1/saml/consume)
2. Encode that authorization request
3. Construct a URL of entryPoint + SAML request in query param
    a. (https://redowl.onelogin.com/trust/saml2/http-post/sso/533049)
4. Redirect to the above URL

SAML is an XML-based protocol and the contents are base64 encoded. If you are using a big string of text like "nVNN....2F%2Fxvk%2F", you will need to decode (and possibly decrypt) the string using a tool like SAMLTool or SSOCircle.

The SAML Test Connector app in OneLogin, which was used to develop the SAML SSO integration, allows you to configure settings so that you can integrate a Service Provider (Forcepoint Behavioral Analytics) with OneLogin as an Identity Provider without using OneLogin's production settings. The ACS URL corresponds to the `path` or `callbackURL` configured in the UI (essentially the server route that will handle SAML responses).

Data Movement and Processing

3.3 Physical Architecture



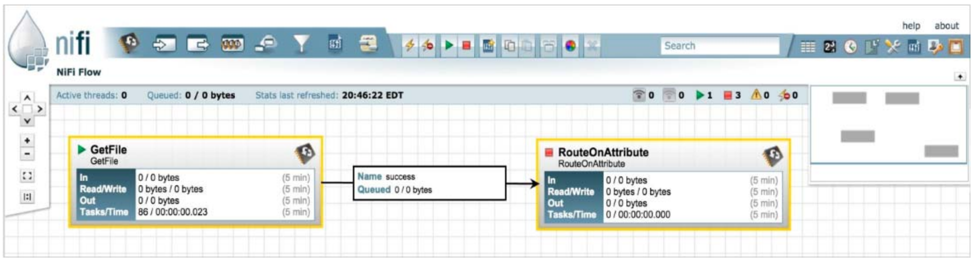
Apache NiFi

For in-depth usage instruction, refer to the official Apache NiFi documentation. We use Apache NiFi for ingesting raw client data into the Forcepoint Behavioral Analytics system, and reliable and secure data transfer between client sources and the Forcepoint Behavioral Analytics Public API. NiFi is also used for data enrichment, preparation of data, conversions between formats, as well as extracting and parsing and routing decisions.

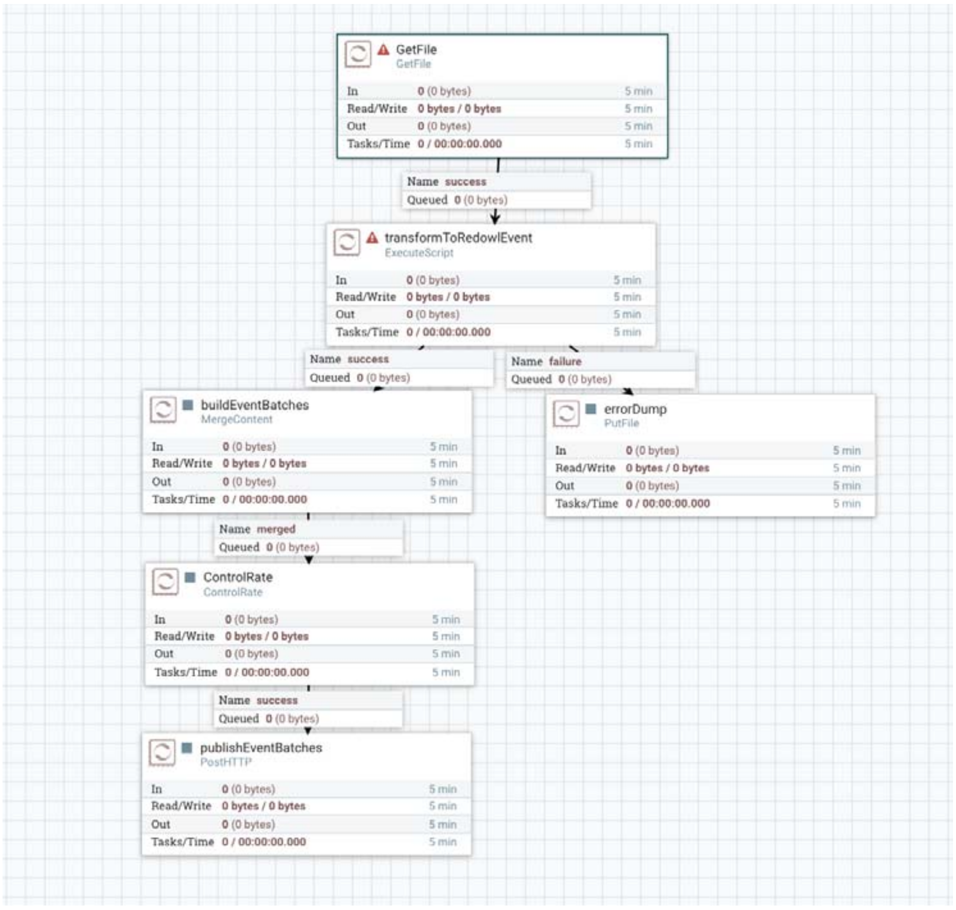
The NiFi system includes a comprehensive user interface allowing you to dynamically configure and execute ingest processes based on client needs. Each NiFi setup includes a "Flow" with the following components:

- FlowFile
  - Unit of data moving through the system
  - Content + Attributes (key / value pairs)
- Processor
  - Performs the work, can access FlowFiles
- Connections
  - Links between processors
  - Queues that can be dynamically prioritized
- Process Group
  - Set of processors and their connections
  - Receive data via input ports, send data via output ports

Flows are configured by dragging and dropping processors:



Example Forcepoint Behavioral Analytics NiFi Flow:



Post Processing

After event data has been moved through each processor in the NiFi flow, it is output to the Public API, the first step in the Streaming Ingest process.

PST Parser

Definition

Not all Forcepoint Behavioral Analytics data is ingested as real-time activity; sometimes historical data is handed to Forcepoint Behavioral Analytics in bulk format. A commonly observed bulk format is a Personal Storage Table or PST file, a file format used by Microsoft software as a compressed message store. The Streaming PST Parser is responsible for uncompressing, determining content type (email, chat, etc.) and delivering the Event data into Streaming Ingest.

Streaming Ingest

Streaming Ingest is the pipeline of procedures used to move properly-formatted data into the Forcepoint Behavioral Analytics data stores. Comprised of five main components, the Forcepoint Behavioral Analytics Streaming Ingest handles both data movement and processing within the pipeline.

1. Public API

The Public API is a web service exposing a public-facing API for publishing data into Forcepoint Behavioral Analytics. The Public API accepts Events and Entities with the following JSON formats. The Public API rejects malformed data with an appropriate error message.

Forcepoint Behavioral Analytics “Event” Format

Field	Required	Type	Sample	Description
type	Yes	string	email	Name of the Event type.
timestamp	Yes	string	2016-01-01T15:45:54	Timestamp for the event (in ISO 8601).
entities	No	EntityRole[ ]	See Entity roles	List of Entities on the Event, with their associated roles.
timezone	No	string	UTC	Timezone in which the Event occured.
source_event_id	No	string	1d4424ee8bf3	A unique identifieer for the event.
subject	No	string	Here’s a subject.	Subject of the Event.
content	No	string	Here lies some content.	Unstructured text associated with the Event.
labels	No	string[ ]	[“interesting”, “unusual”]	Labels to add to the event.

Field	Required	Type	Sample	Description
attributes	No	EventAttribute[ ]	See Event Attributes	List of arbitrary name-value pairs that define Attributes of the Event.
attachments	No	Attachment[ ]	See Attachments	List of attributes to attach to the event.
event_references	No	EventReferences	See Event References	A data structure allowing this Event to be linked to others.

Forcepoint Behavioral Analytics "Event Attribute" Format

Field	Required	Type	Sample	Description
name	Yes	string	encrypted	Name of the Attribute.
value	Yes	string	true	Value of the Attribute.
type	Yes	string	boolean	Type of the Attribute. Valid values: string, boolean, double, date. Defaults to string.

Entity Role Format

Field	Required	Type	Sample	Description
role	Yes	string	src_ip	The given Entity's role in the Event.
entities	Yes	string[ ]	["12.34.56.4789]	List of entities.

Usage

Events can be loaded one-by-one or in bulk. The following "cURL" command is an example, including the parameters needed to send Events using the Public API.

Loading an event:

```
curl -XPOST localhost:9000/event @event.json -H content-
type:application/json
```

Request

Header:

```
Content-Type: application/json
```

Body:

```
{
  "type": "irc-message",
  "timestamp": "2016-01-01T15:45:54",
  "subject": "notification to #Forcepoint",
  "content": "Added a new channel.",
  "entities": [
    {
      "role": "from",
      "entities": [
        "foo123"
      ]
    },
    {
      "role": "to",
      "entities": [
        "bar456",
        "baz789"
      ]
    }
  ],
  "attributes": [
    {
      "type": "string",
      "name": "channel",
      "value": "#Forcepoint_news"
    }
  ]
}
```

Usage

Loading bulk events:

```
curl -XPOST localhost:9000/event/bulk --data-binary
@bulk_events.json -H content-type:application/x-json-stream
```

Request

Header:

```
Content-Type: application/json
```

Body:

```
{"type": "irc-message", "timestamp": "2016-01-01T12:00:00",
"content": "want to grab some bbq?", "entities": [ {"role":
"from", "entities": ["foo123"]}, {"role": "channel",
"entities": ["#water_cooler"]}]}]]

{"type": "irc-message", "timestamp": "2016-01-01T12:00:01",
"content": "want to grab some bbq?", "entities": [ {"role":
"from", "entities": ["foo123"]}, {"role": "channel",
"entities": ["#water_cooler"]}]}]]

{"type": "irc-message", "timestamp": "2016-01-01T12:00:02",
"content": "want to grab some bbq?", "entities": [ {"role":
"from", "entities": ["foo123"]}, {"role": "channel",
"entities": ["#water_cooler"]}]}]]
```

2. RabbitMQ

RabbitMQ is an integral part of the Streaming Ingest service for Forcepoint Behavioral Analytics that stores incoming Events and metadata between services waiting to be processed. RabbitMQ consists of messages, queues, and exchanges.

- Messages are the data that's moved through the services.
- Queues are where messages get lined up for impending exchanges (movement out of RabbitMQ).

Configuration of RabbitMQ server via rabbitmq.config

The RabbitMQ server has many configuration parameters that can be set in rabbitmq.config; the following three are most essential to the health of the RabbitMQ server. A full list of parameters can be found in https://www.rabbitmq.com/configure.html.

Note

Default values are used for anything not explicitly changed or set in the configuration document.

Parameter	Value	Default	Description
vm_memory_high_watermark	0.6	0.4	Memory (smaller of installed RAM or virtual address space) fraction threshold at which flow control is triggered. When the memory threshold is hit then publishers begin to be throttled. A queue that is being throttled will show "flow" as its state.
vm_memory_high_watermark_paging_ratio	----	0.5	Fraction of vm_memory_high_watermark at which queues start to page messages out to disc to free up memory.
disk_free_limit	----	50 MB	Minimum free space available before flow control is triggered in order to avoid a server crash due to insufficient space. Note that if messages are being paged out rapidly it is possible to run out of disk space and crash in the time between two runs of the disk space monitor.

RabbitMQ Configuration Options

The following options can be configured for each one of the Forcepoint Behavioral Analytics services. In the table below, the column "Forcepoint Behavioral Analytics Default" refers to the values used when creating the RabbitMQ ConnectionFactory.

The column "RabbitMQ API Default" refers to the default value used in RabbitMQ's API.

Parameter	RabbitMQ API Default	Forcepoint Behavioral Analytics Default	Description
prefetchCount	unlimited	10	Max number of unacknowledged messages in channel/connection. See https://www.rabbitmq.com/blog/2012/05/11/some-queuing-theory-throughput-latency-and-bandwidth/ for a nice description of the tradeoffs in setting prefetch count.
prefetchLimitGlobal	N/A	true	prefetchCount settings are applied to the entire channel rather than each consumer
requestedChannelMax	0	0	Maximum permissible number of channels to negotiate with clients. Setting to 0 means "unlimited". Using more channels increases memory footprint of the broker.
requestedFrameMax	0	0	Maximum permissible size of a frame (in bytes) to negotiate with clients. Setting to 0 means "unlimited" but will trigger a bug in some QPid clients. Setting a larger value may improve throughput; setting a smaller value may improve latency.

Parameter	RabbitMQ API Default	Forcepoint Behavioral Analytics Default	Description
requestedHeartbeat	0	0	Time in seconds after which, if no response is received, the TCP connection is considered unreachable and reconnection is needed. Zero means that the checking is not performed. Values between 5 and 20 seconds are optimal for most environments according to http://www.rabbitmq.com/heartbeats.html.
connectionTimeout	0	0	Wait indefinitely for Events to get in the queue.
automaticRecovery	false	false	Recovery of connections. Note that newer versions of RabbitMQ set this to TRUE by default.
topologyRecovery	true	true	Recovery of exchanges, queues, bindings and consumers.
networkRecoveryInterval	5000 ms	5000 ms	Retry time interval after a recovery attempt failure.

Create the RabbitMQ Queues, Exchanges and Bindings

Invoking the "create" directive for pipeline-ctl will create the following Entities in RabbitMQ.

Queue	Exchange	Routing Key (Binding)
elasticsearch.error.queue	elasticsearch.error.exchange	error
elasticsearch.event.queue	elasticsearch.event.exchange	event
reveal.error	reveal.error.exchange	reveal.error
reveal.event	reveal.event.exchange	reveal.json

Queue	Exchange	Routing Key (Binding)
reveal.internal.error	reveal.internal.error.exchan ge	reveal.internal.error
reveal.internal.event	reveal.internal.exchange	reveal.model

After you invoke the following command, then you can verify the queues, exchanges, and Routing Keys in the RabbitMQ Admin Console. Create RMQ Queues, Exchanges, and Bindings.

```
sudo pipeline-ctl --rmq-create
```

Scenario: RabbitMQ server crashes or network issues

Effect on API, Conversion, and Queue Worker Services

If the RabbitMQ server crashes or becomes unreachable due to network problems, reveal-public-api will stop accepting messages and will respond to any new request with:

```
{
  "status": 500,
  "message": "Cannot reach queue.",
  "cause": "Failed to create a RabbitMQ channel",
  "exception": "javax.ws.rs.WebApplicationException",
  "stackTrace": [...]
}
```

At the same time, the conversion and Queue Worker services are automatically paused until the connection to the RabbitMQ server is re-established. For example, the Queue Worker log will show:

```
WARN  [2017-02-10 19:17:47,191]
com.redowlanalytics.reveal.ingest.core.InternalEventQueueRea
der: Connection to queue is down - waiting to reconnect…
```

Scenario: Consumer Crashes

Effect on Logstash

Ensure you have sufficient disk space on the Rabbit server, so it can still buffer even when a consumer is temporarily not running. The disk_free_limit parameter controls when it will start to block writes. When the storage falls below the "disk_free_limit" parameter, everything will be blocked.

Disaster Recovery

In order to maintain service availability in the event of disk failures, RabbitMQ needs to be configured as a cluster (see https://www.rabbitmq.com/ha.html). With a cluster, the queues, exchange, and bindings can be mirrored. This ensures that if any of the master nodes were to fail, one of the slave nodes would take its place maintaining continuous delivery of the messages.

3. Conversion

Events in the external Event queue are converted into the Forcepoint Behavioral Analytics internal data model through the Conversion web service. This includes JSON events from the public API and RFC-822 email from the Public API.

A chat Event is usually composed of multiple chat messages concatenated into one. Forcepoint Behavioral Analytics applies a chat-splitting procedure to separate concatenated chat Events into individual messages for analysis. The Conversion API modifies the chat-splitting behavior by submitting custom configuration settings or retrieving existing settings.

Non-chat Events are simply converted into Forcepoint Behavioral Analytics JSON format. There is no additional processing required.

How it works

When parsing chat data using Reveal ETLE CLI, you need the following items:

- A data source where chat content is stored in the body of individual emails
- A way to examine the contents of the email/chat events before fully ingesting
- A JSON config file

1. Examine the content.

   Either run ETLE on a subset of data and look at the results in the UI, or use a tool to extract and view the events first.

   a. Determine Single or Multi-Line Chat Format:

      Determine whether you would like your chat fit on one, or multiple lines.

      Sample single-line chat:

      ```
      >>>>02/10/2015 14:33:22    JDOE1, JIM DOE, SOME
      COMPANY Says:Yo, let's do lunch
      ```

      Sample multi-line chat:

      ```
      # Participant InstantBloomberg:somebb3 entered on Jun
      1, 2015 2:12:17 AM
      # Participant information
      buddyName: somebb3
      networkID: InstantBloomberg
      # End of participant information
      InstantBloomberg:somebb3 (2:12:18 AM):
      *** FOO BAR INC(CANADA)LTD (11111) Disclaimer:
      disclaimer text here
      InstantBloomberg:somebb3 (2:13:11 AM):
       Is it lunch time?
      ```

   b. Determine How Many Chat Types You'll Have

      Look for all the chat you might get. Search (case insensitive) for things like "conversation", "jabber", "bloomberg", "lync", "chat", "joined", etc.

2. Build Your JSON Config File.

See the sample request below the Configuration File Properties for a file configuration example.

3. Run ETLE CLI.

   Run reveal-etle-cli with the -c flag:

```
reveal-etle-cli ... -c /path/to/config.json
```

Setting Chat Splitting Configuration

You can configure chat splitting by submitting a configuration file with a POST request. You can retrieve the current chat splitting configuration with a GET request. The configuration file contains one or more "chat_config" sections. The table below summarizes the fields supported in the chat_config sections.

Configuration File Properties

Field	Type	Example	Notes
name	string	“BB Chat: Single-Line body Format”	Not technically used. For JSON readability only.
enabled	boolean	“true”	
multi-line	boolean	“false”	Determines whether chat messages are displayed with line breaks- useful for readability.
chat_patterns	object[]	See chat patterns table.	
mode_support	object[]	See Mode Support table.	

Chat Patterns

Field	Type	Example
regex	string	“[*]{15}Start message properties[*]{15}.*RoomID: ”
event_field	boolean	“content”

Mode Support

Field	Type	Example
identifier	string	“disc”
ingest_enabled	boolean	“false”

Field	Type	Example
event_type	string	“bloomberg-disclaimer”
type_detection_regex	string	“[>](\d{2}\/\d{2}\/\d{4} \d{2}:\d{2}:\d{2})[ ]+(.) Says:[ ]+[*]+.Disclaimer: (.)”
date_regex_group	integer	1
sender_regex_group	integer	2
body_regex_group	integer	3
date_format	string	“MM/dd/yyyy HH:mm:ss” i.e. based on Java SimpleDateFormat
default_timezone_id	string	“UTC”

Accessing and Changing Chat Splitting Configurations

Chat Events often enter the Public API as emails with concatenated chat Events as the email body. Chat Splitting detects chat format in an email body, separates out the individual chat messages, and outputs an Event for each individual chat.

Request

Header:

```
Content-Type: application/json
```

Body:

```
{
  "chat_config": [
    {
      "name": "BB Chat: Single-Line Body Format",
      "enabled": "true",
      "multi_line": "false",
      "chat_patterns": [
        {
          "regex": "[*]{15}Start message 
properties[*]{15}.*RoomID:",
          "event_field": "content"
        },
        {
          "regex": "Bloomberg.*",
          "event_field": "X-KVS-MessageType"
        },
        {
          "regex": "(CHAT-fs:)|(PCHAT-0x)",
          "event_field": "subject"
        }
      ],
      "mode_support": [
        {
          "identifier": "disc",
          "ingest_enabled": "false",
          "event_type": "bloomberg-disclaimer",
          "type_detection_regex": "[>]*(\\d{2}\\/\\d{2}\\/
\\d{4} \\d{2}:\\d{2}:\\d{2})[ ]+(.*) Says:[
]+[*]+.*Disclaimer: (.*)",
          "date_regex_group": 1,
          "sender_regex_group": 2,
          "body_regex_group": 3,
          "date_format": "MM/dd/yyyy HH:mm:ss",
          "default_timezone_id": "UTC"
        },
        {
          "identifier": "enter",
          "ingest_enabled": "false",
          "event_type": "bloomberg-enter",
          "type_detection_regex": "[>]*(\\d{2}\\/\\d{2}\\/
\\d{4} \\d{2}:\\d{2}:\\d{2})[ ]+(.*) (has joined the room)",
          "date_regex_group": 1,
          "sender_regex_group": 2,
          "body_regex_group": 3,
          "date_format": "MM/dd/yyyy HH:mm:ss",
          "default_timezone_id": "UTC"
        },
        {
          "identifier": "exit",
          "ingest_enabled": "false",
          "event_type": "bloomberg-exit",
          "type_detection_regex": "[>]*(\\d{2}\\/\\d{2}\\/
\\d{4} \\d{2}:\\d{2}:\\d{2})[ ]+(.*) (has left the room)",
          "date_regex_group": 1,
          "sender_regex_group": 2,
          "body_regex_group": 3,
          "date_format": "MM/dd/yyyy HH:mm:ss",
          "default_timezone_id": "UTC"
        },
        {
          "identifier": "msg",
          "ingest_enabled": "true",
          "event_type": "bloomberg-chat",
          "type_detection_regex": "[>]*(\\d{2}\\/\\d{2}\\/
\\d{4} \\d{2}:\\d{2}:\\d{2})[ ]+(.*) Says:[
]+(?!.*(?>\\*\\*\\*.*Disclaimer))(.*)",
          "date_regex_group": 1,
          "sender_regex_group": 2,
          "body_regex_group": 3,
          "date_format": "MM/dd/yyyy HH:mm:ss",
          "default_timezone_id": "UTC"
        }
      ]
    },
    {
      "name": "BB Chat: Multi-Line Body Format",
      "enabled": "true",
      "multi_line": "true",
      "chat_patterns": [
        {
          "regex": "Bloomberg.*",
          "event_field": "X-KVS-MessageType"
        },
        {
          "regex": "Bloomberg IM Conversation #",
          "event_field": "subject"
        }
      ],
      "line_join_delimiter_regex": [
        "^# Interaction information",
        "^# Start of interaction",
        "^InstantBloomberg:",
        "^# Participant InstantBloomberg:",
        "^# Interaction information",
        "^# End of participant information"
      ],
      "mode_support": [
        {
          "identifier": "msg",
          "ingest_enabled": "true",
          "event_type": "bloomberg-chat",
          "type_detection_regex":
"^InstantBloomberg:([\\w]+).*\\((\\d{1,2}:\\d{2}:\\d{2}
[AP]M)\\):(?!.*\\*\\*\\*.*Disclaimer)(.*)",
          "date_regex_group": 2,
          "sender_regex_group": 1,
          "body_regex_group": 3,
          "date_format": "hh:mm:ss a",
          "default_timezone_id": "UTC"
        }
      ]
    },
    {
      "name": "Jabber Chat: Multi-Line Body Format",
      "enabled": "true",
      "multi_line": "true",
      "chat_patterns": [
        {
          "regex": "ChatRoom:Jabber",
          "event_field": "X-FaceTime-IMA-buddyName"
        },
        {
          "regex": "Jabber Conversation #",
          "event_field": "subject"
        }
      ],
      "line_join_delimiter_regex": [
        "^[^\\t@]+"
      ],
      "mode_support": [
        {
          "identifier": "msg",
          "ingest_enabled": "true",
          "event_type": "jabber-chat",
         "type_detection_regex": "^(\\d{2}\\/\\d{2}\\/\\d{2}
\\d{1,2}:\\d{2}:\\d{2} [AP]M)\\t
(.*)\\t@(.*)\\t(?!(Left|Joined) conversation\\.)(.*)",
          "date_regex_group": 1,
          "sender_regex_group": 2,
          "body_regex_group": 5,
          "date_format": "MM/dd/yyyy hh:mm:ss a",
          "default_timezone_id": "UTC"
        }
      ]
    },
    {
      "name": "Lync: Single-Line Body Format",
      "enabled": "true",
      "multi_line": "false",
      "chat_patterns": [
        {
          "regex": "MicrosoftUC\\.IM",
          "event_field": "X-FaceTime-IMA-contentType"
        },
        {
          "regex": "IMChat MicrosoftUC Conversation #",
          "event_field": "subject"
        }
      ],
      "mode_support": [
        {
          "identifier": "msg",
          "ingest_enabled": "true",
          "event_type": "lync-chat",
          "type_detection_regex": "^(\\d{4}-\\d{2}-
\\d{2}T\\d{2}:\\d{2}:\\d{2}[+-]\\d{4})\\t \\t(.*)
\\t(?!(Left|Joined) conversation\\.)(.*)\\s\\t$",
          "date_regex_group": 1,
          "sender_regex_group": 2,
          "body_regex_group": 4,
          "date_format": "yyyy-MM-dd'T'HH:mm:ssZ",
          "default_timezone_id": "UTC"
        }
      ]
    },
    {
      "name": "Placeholder: Just here for kicks",
      "enabled": "false",
      "multi_line": "false"
    }
  ]
}
```

4. Queue Worker

The Queue Worker applies a variety of Event transformations and analytics to Events being streamed into Forcepoint Behavioral Analytics. The Queue Worker pipeline is composed of multiple processors. Each processor performs a specific task to a single incoming Event and passes that Event along to the next processor in the pipeline.

Pipeline behavior can be configured through the UI and / or API by adding, removing, or reconfiguring the following processors:

1. Deduplication

The Deduplication Processor generates the (Elasticsearch document) ID for each Event. The ID is a hash of the fields specified in the processor configuration. Note that the specified deduplication fields are those from the internal Event Model (typically, the fields available in the stored Elasticsearch documents).

Configuration Properties

Field	Type	Description
type	sting	Must be deduplication.
fields	string[]	List of Event fields to use for deduplication.
timestamp_granularity	string	Events that occur within the same granularity are considered to have equivalent timestamps for the purposes of deduplication.
timestamp_time_zone	string	Indicate the timezone that corresponds to a different day for timestamp_granularity of that level.

Available Fields:

- source_event_id
- type
- timestamp
- entities
- subject
- content
- attributes
- attachments

  Example:

```
{
  "type": "deduplication",  "fields": [ "timestamp",
"entities", "attributes", "source_event_id" ]
}
```

The Queue Worker's Deduplication Processor accepts two additional configuration settings: "timestamp_granularity" and "timestamp_time_zone"

The user can now pick a granularity (e.g., "day") and events that occur within the same day are considered to have equivalent timestamps for the purposes of deduplication. The "timestamp_time_zone" controls when the boundaries of those time windows occur (e.g., when the "start of day" happened). The default timestamp_granularity is milliosecond (expressed as ms).

2. Entity Resolution

The Entity Resolution Processor automatically resolves the raw value(s) in an Event's Entity fields to a known Entity in the Forcepoint Behavioral Analytics platform. The Forcepoint Behavioral Analytics Master Data Service (MDS) maintains the process of mapping raw values to known Entities. This processor periodically retrieves this mapping from Elasticsearch. The processor resolves the roles field for each Event. If any given value cannot be associated with a known Entity, a new Entity is added to MDS's reference data.

Configuration Properties

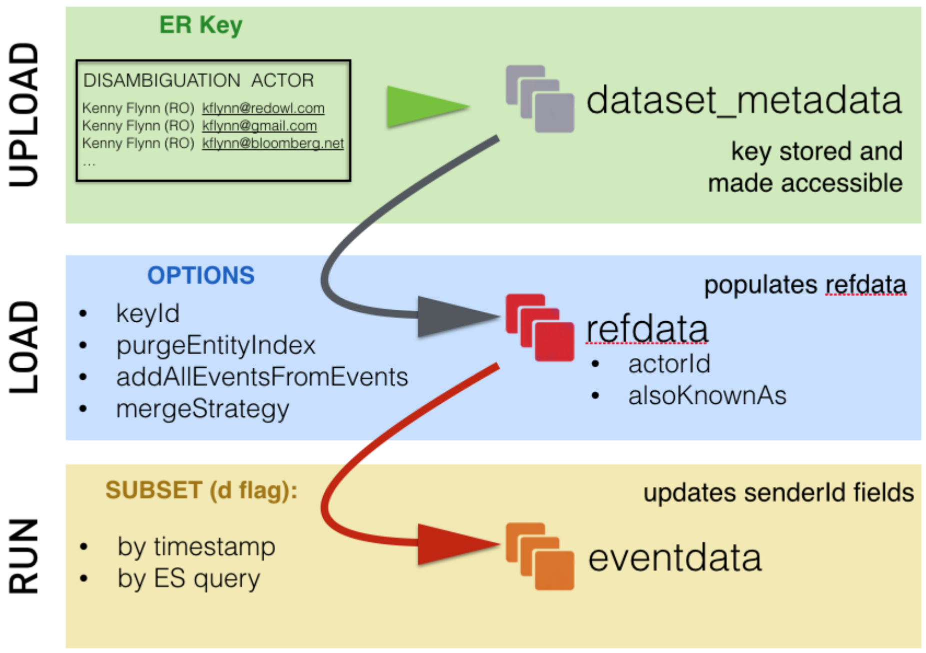
Field	**Type**	**Description**
type	string	Must be entity_resolution.
key_refresh_period	string	A time value in NUMBER unit format, allowable units are **ms** for milliseconds, **s** for seconds , and **m** for minutes (**default 15m**)
new_actor_publish_period	string	A time value in NUMBER unit format, allowable units are **ms** for milliseconds, **s** for seconds , and **m** for minutes (**default 2m**)
use_smtp_addresses	boolean	Overwrite all email addresses with valid extracted smtp address if one can be found, e.g., “Paul <paul@redowl.com>” will be converted to “paul@redowl.com” wherever it is seen. (default false)

Field	Type	Description
keep_smtp_entities_as_attributes	boolean	Save the original email address as an event attribute, when it is replaced by an extracted smtp address. (default false)
update_entity_dictionary_lock_timeout_period	string	A time value in NUMBER unit format, allowable units are **ms** for milliseconds, **s** for seconds , and **m** for minutes (**default 1m**)

Example:

```
{
  "type": "entity_resolution",
  "key_refresh_period": "5m",
  "new_actor_publish_period": "2m",
  "use_smtp_addresses": false
}
```

How Entity Resolution Works



Maintaining Entity Resolution

1. Upload the New Key

```
curl -k -XPOST -F "file=@/path/to/
disambiguation.key;type=text/csv" https://localhost:8080/
reference/disambiguation/upload
```

Note

The “@” sign is not a suggestion and is REQUIRED. This symbol tells cURL to look for this content inside a file at the following path.

The “;type=text/csv” is also required and establishes what dataset_metadata should expect for the format.

If successful, you will get a response with a keyID you will need to use in the subsequent LOAD step.

Testing this step:

```
curl -k -u elastic:changeme https://localhost:9200/
dataset_metadata/entitydisambiguation/
AVJqiBa70P1oKZcqlJac | jq .
```

2. Load using the desired parameters

```
curl -k -XPOST 'https://localhost:8080/reference/
disambiguation/
load?keyId={MYKEYID}&addAllEntitiesFromEvents=true&purgeE
ntityIndex=false&mergeStrategy=KEY_WINS'
```

Options:

- addAllEntitiesFromEvents: true or false - this adds into refdata all Entities in the dataset, even if they are not resolved in the key
- purgeEntityIndex: true or false - this wipes out refdata and starts fresh
- mergeStrategy: KEY_WINS or INDEX_WINS - in this case, merging priority goes to the key. This means that if there is conflict between the key and refdata, a new document will be created in refdata to reflect what is in the key

The key is now in refdata. This means that streaming ingest will now point to the updated key during ingest. This may take 5-10 minutes to take effect. In order to test the quality of refdata, queries may be run in the following format:

```
curl -k -u elastic:changeme https://localhost:9200/
refdata/_search -
d'{"query":{"query_string":{"query":"Flynn"}}}' | jq
.hits.hits[]
```

3. Reload Entity Attributes using new ActorID (if applicable).

   See Attributes in Forcepoint Behavioral Analytics below.

4. Run Entity Resolution over past Events.

   Run Entity resolution on all the Events with that ActorID as a sender or recipient on a weekly basis.

```
curl -k -XPOST https://localhost:8080/reference/
disambiguation/run
```

```
-d '{"query": {"bool": {"should": [{"match": {"sender":
"Smith, Jane (jSmith)"}}, {"match": {"sender": "Brown,
Joe (jBrown)"}}, {"match": {"recipient": "Smith, Jane
(jSmith)"}}, {"match": {"recipient": "Brown, Joe
(jBrown)"}}]}}}'
```

Track job progress:

```
curl http://localhost:9200/jobs/disambiguation/jobId | jq
.
```

5. Maintain Changelog.

   A changelog should be maintained on a weekly basis to show the delta of the changes to the ER key. For example:

Actor	Disambiguation	Changed By	Date
Smith, Jane (jSmith)	jSmith@mlp.com	Johnson, Jack	October 10, 2016
Brown, joe (jBrown)	jBrown@mlp.com	Johnson, Jack	October 10, 2016

Other jobs

Entity Resolution can be run on a larger subset of the data, at the discretion of the team. For example, if phone numbers are resolved in a new key, ER will need to be run over the weekend over all blackberry data.

```
curl -XPOST http://localhost:8080/reference/disambiguation/
run
-d '{"query": {"match": {"type": "blackberry-email"}}}'
```

Attributes

Inserting attributes

The script entitled addActorPropertiesNewModel.py has been updated to support startTime and endTime fields in milliseconds. This procedure requires that the actor already exist in refdata. If the actor is not in refdata, the post command will return an error.

The command to run is as follows:

```
python addActorPropertiesNewModel.py http://localhost:8080/
reference/actor --post attributes.csv
```

Please change "http://localhost" to the Forcepoint Behavioral Analytics host, and "attributes.csv" to the name of the file in which the Attributes have been listed (see sample below).

Sample CSV (attributes.csv):

Actor,AttributeName,AttributeValue,startTime,endTime

"Smith, Jane (jSmith)", Department, Information Systems, 1451682809000, 1483305209000

Note that the start and end time are UTC timestamps in milliseconds.

Post return:

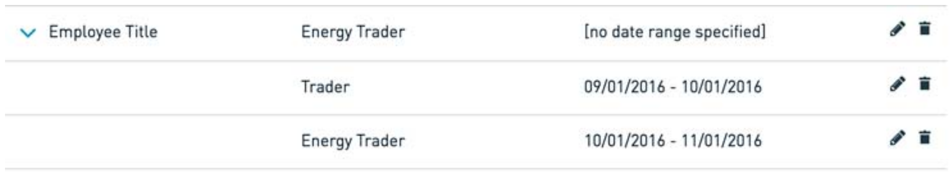
```
$ python AddActorProperties.py mds1:8080/reference/actor --
post attributes.csv
```

FIELD: Employee title interpreted as type string with format None

```
curl -XPOST -H 'Content-Type: application/json' -d
'{"startTime": 1451682809000, "endTime": 1483305209000,
"name": "Department", "value": "Information Systems"}'
'mds1:8080/reference/actor/Smith%2C%20Jane%20(jSmith)/
attribute/string'
```

Inserting Attributes where the Attribute value is the same but the dates are different

Using the script above, it is now possible to use insert Attribute values with new start and end times.



Employee Title	Energy Trader	[no date range specified]	
	Trader	09/01/2016 - 10/01/2016	
	Energy Trader	10/01/2016 - 11/01/2016	

The post command will require that one of the values be different - that is, if the exact same Attribute Name, Value and date range is posted, it will return an error. If any of the values is different, it will post the new entry to the endpoint, which will display as above.

Posting several attributes at once

When the script is run, each row of the attributes CSV is posted to the endpoint. Each row that has new values will post successfully. If a duplicate row is posted, the endpoint will return an error and move on to the next row.

Updating and deleting attributes

This functionality is not supported using the script provided above. Future releases of ROSE will support updating and deleting attributes.

3. Detect Disclaimers

The Detect Disclaimers processor classifies disclaimer paragraphs. Specifically, it counts the percentage of n-grams in a paragraph that also appear in a disclaimer lexicon. If this percentage exceeds the provided threshold, then the paragraph is marked as a disclaimer. Disclaimers may be taken into account for downstream analytics and in-app visualizations.

Configuration Properties

Field	Type	Description
type	string	Must be detect_disclaimer.
lexicon_id	string	The Elasticsearch ID of the disclaimer n-gram Lexicon. The Lexicon must have type WORD_NGRAMS.
threshold	double	Minimum percentage of paragraph n-grams that must appear in disclaimer Lexicon. Between 0.0 and 1.0; recommend setting to 0.4.

Example:

```
{
    "type": "detect_disclaimer",
    "lexicon_id": "AVYtKdRBDjvQ3tDtq7uW",
    "threshold": 0.4
}
```

4. Feature Scoring

The Feature Scoring processor is an internal processor that extracts Feature values from incoming Events and assigns a score to each of the Features. This processor is implicitly configured by the Feature configuration page within Forcepoint Behavioral Analytics. Modifications in-app to those Features are automatically propagated to the ingest pipeline.

Example:

```
{
  "type": "feature_scoring",
  "feature_extractors": [
    {
        "type": "time_grouping_boolean",
        "feature_id": "AVeRgRm5aNb9iwicESzx",
        "name": "Time Grouping Feature",
        "time_grouping": "QUARTER_OF_YEAR",
        "values": [1]
    }
  ]
}
```

To configure a categorical event attribute feature, select only the attribute name among all event attributes of type string, taking into consideration that they also have the ability to do so with entity roles.

By default, String Categorical Features are configured to score on the top-100k categories. This configuration parameter, `max_bins`, is not exposed during the feature creation through the UI.

A soft-limit will be applied to the number of bins stored. Once the soft-limit is reached, no more bins are added.They are instead added to an overflow bin.

In a multi-queue worker environment, each queue worker will have its own version of the container. These containers are merged periodically, sometimes resulting in a new container with more bins than the limit. This is OK, as after container synchronization, no more new values would be added. The probability of a value that falls in the overflow bin is always null and they are at the end of the values list.

5. Queue Worker Enrichment Processor for Attachment Number and Attachment Byte Count

As of version 2.60 of the Forcepoint Behavioral Analytics, attachment related information like the total number of attachments and the sum of the attachment sizes are now added to the event as EventAttributes of type double named Attachment Count and Total Attachment Bytes, respectively. These new attributes are created by default if at least one attachment is present in the event. This behavior can be disabled by clients/OPS by posting the following processor configuration to the queue-worker processor endpoint:

```
{
    "type": "attachment_enrichment",
    "enabled": false
}
```

The names Attachment Count and Total Attachment Bytes are now "reserved", hence, no attributes with those names can be created via the API.

Historic events will be migrated to reflect this addition. Existing EventAttribute of type double with the above names will NOT be overwritten during migration.

The default queue-worker includes this enrichment processor enabled.

6. Queue Worker Enrichment Processor for Classifying Sender and Receivers

As of Forcepoint Behavioral Analytics version 2.60, the Queue Worker adds the ability to classify the entities associated with a given role as being part of one or more local domains or exchange organizations. If all, none, or some entities in the role match one of the domains a new attribute (sender Role Domain or recipient Role Domain) is created with value of "All Internal", "All External", or "Mixed", respectively.

Entities that can't be matched to any domains or organizations are considered external.

For example, the following configuration will match the entities in the roles “sender” and “recipient” to the local domain “bar.com”.

```
{
 "type": "entity_domain_enrichment",
 "enabled": true,
 "roles": ["sender", "recipient"],
 "internal_domains": ["bar.com"]
}
```

With the above enrichment processor in place an event containing a role like:

```
{
 "role": "sender",
 "entities": ["alice@bar.com", "bob@foo.bar.com"]
}
```

would result in the attribute "sender Role Domain" with value "All Internal". If "bob@foo.bar.com" in the previous role is replaced with "bob@foobar.com" then the value of the attribute would become "Mixed" since "foobar.com" is not a sub-domain of "bar.com".

An event with a role like:

```
{
 "role": "sender",
 "entities": ["alice", "bob"]
}
```

would result in an attribute value of "All External" since none of the entities in the entities in the role can be match to the "bar.com" domain.

Moreover, it is possible to configure the entity domain enrichment processor for the resulting attributes:

```
{
   "type": "entity_domain_enrichment",
    "enabled": true,
    "roles": ["sender", "recipient"],
    "matching_domains": ["edu"],
    "matching_group_name": "Edu",
    "nonmatching_group_name":"Non-Edu"
    "mixed_group_name":"Mixed-Edu"
}
```

would result in attributes with name “sender Role Domain” with values of “Edu” or “Non-Edu”.

Must ensure multiple configurations don't collide (e.g., "in_domain_name" can't be "Edu" for two different processor configs).

7. Prepare for Elasticsearch

The Elasticsearch Preparation processor is an internal helper that adds some necessary storage metadata to processed Events so external services know how to index those Events into Elasticsearch. Specifically, it adds the following fields to an Event just before publishing to the message queue: doc_index, doc_type, doc_id. Downstream consumers of these messages are responsible for handling these additional Event fields.

Example:

```
{
  "type": "elasticsearch_preparation"
}
```

Data Storage and Querying

Overview

RabbitMQ: acts as an intermediary queue for Forcepoint Behavioral Analytics data as it awaits interaction with the following services:

- Public API Service
- Conversion Service
- Queue worker processing
- Logstash transfer to MDS

Service	Description
Logstash	The service that transfers data between RabbitMQ and Elasticsearch. Once data is output from the Queue Worker, Logstash grabs Event, Entity, and metadata from RabbitMQ and pushes it to Elasticsearch.
Elasticsearch	Elasticsearch is an efficiently searchable, enterprise-grade search engine that stores all Forcepoint Behavioral Analytics Events. Data in Elasticsearch is queried through MDS.

Service	Description
PostgreSQL	The data-store responsible for holding relational data which primarily includes user account information, entitlements, authorization, user preferences, notifications and some analytics information. The Ansible install delivers a fully configured PostgreSQL, so no further configuration is necessary.
MDS	The service that queries data from Elasticsearch and performs analytics before sending the results to the UI for display to Forcepoint Behavioral Analytics users.

Master Data Service

The Master Data Service (MDS) is the Forcepoint Behavioral Analytics API for querying data from Elasticsearch to the UI for analytics and user interaction.

While diving into MDS configuration can be risky and only meant for the power user, it is configurable through the AppConfig JSON document. Variables, descriptions, and example values can be found in the Appendix.

Data Management

Timeslicing

Motivation

Historically, Forcepoint Behavioral Analytics kept all customer Events in a single index called eventdata in Elasticsearch. In order to improve performance, Forcepoint Behavioral Analytics introduced additional time-based indices.

While the index creation frequency is configurable via "timeslice_window_period": "week" OR "month", it is advised that week-based indices are used for high volume customers. For instance, if there was an Event created on July 2016, then Forcepoint Behavioral Analytics will generate indices of the form eventdata_YEAR-WeekNum (eg. eventdata_2016-29, corresponding to the week of July 18). Month-based indices are used by default in the Forcepoint Behavioral Analytics V2.50.0 release and beyond.

Update the Configuration

First you need to update the timeslicing configuration by invoking this MDS API endpoint: /reference/config/mds

```
curl -XPUT 'http://<mds-host>:8080/reference/config/mds' -d
'{
    "timeslice_enabled": true,
    "timeslice_archive_enabled": true,
    "timeslice_window": 52,
    "timeslice_window_period": "week"
  }'
```

Test timeslicing by executing this MDS API endpoint: timeslicing/sample

```
curl -XPOST -H "Content-Type:application/json" 'http://
localhost:8080/timeslicing/sample' -d "{}"
```

Note

When updating the timeslicing configuration, note that the job will not re-slice previously ingested data. The updated timeslicing configuration slices ingested data from the updated configuration point forward.

Minigator

The term Minigator is an internal Forcepoint anagram for migration. The Minigator is a command line tool used to migrate data during Forcepoint Behavioral Analytics upgrades, while untangling data migrations in Elasticsearch from the rest of the application. Uses include:

- Data migration
  - Eventdata - data about events
  - Refdata - entity data
  - Metadata - feature, lexicon, etc. data
- Untangling Elasticsearch data
- Search for necessary updates, pull documents, execute change, then re-insert documents
- Write documents from a cluster to the file
- Write documents from a file to the cluster

Instructions for specific usages can be found in the 2.50 Release Notes.

Format

Minigator uses YAML due to its excellent read- and write-ability:

```
# Note: index names and types are examples only
input:
    indices: ["index1", "index2"]
    types: ["document", "error_message"]
    filter: {exists: {field: "creation_date"}}
```

Our migration list can be a simple list of migration names or used to add parameters:

```
# example migration list
migrations:
         - name: "TruncateUsernames"
         - name: "RemoveCreationDateField"
```

Minigator Configuration File Format

The location of the Mingator configuration file is specified using the '--conf' option. Configuration files may specify elasticsearchInput and elasticsearchOutput as seen below.

Elasticsearch Input

```
elasticsearchInput:
  host: "localhost"
  port: "9300"
  cluster.name: "redowl"
  passThroughSettings:
    client.transport.ping_timeout: 10s
  scroller:
    pageSizePerShard: "50"
    scrollTime: "60s"
```

- "host": DNS or IP hostname of the target Elasticsearch cluster.
- "port": port for the Elasticsearch *transport client* (note: this is different from the HTTP port).
- "cluster.name": Name of the cluster.
- "passThroughSettings": a map of any other settings that need to be passed to the Elasticsearch client. See Elasticsearch Documentation for options.
- Scroller
  - "pageSizePerShard" Number of results to return in each scroll result, per shard.
  - "scrollTime": Time to leave the scroller open.

Elasticsearch Output

```
elasticsearchOutput:
  host: "localhost"
  port: "9300"
  cluster.name: "redowl"
  passThroughSettings:
    client.transport.ping_timeout: 10s
  bulkWriter:
    bulkActions: 50
    bulkSize: "5MB"
    bulkCloseTimeout: 30
```

```
        bulkCloseTimeUnit: "SECONDS"
        flushIntervalSeconds: 10
        concurrentBulkActionRequests: 1
```

- “host”: DNS or IP hostname of the target elasticsearch cluster.
- “port”: Port for the elasticsearch *transport client* (note: this is different from the HTTP port).
- “cluster.name”: Name of the cluster.
- “passThroughSettings”: A map of any other settings that need to be passed to the Elasticsearch client. See Elasticsearch documentation for options.
- Bulk Writer
  - “bulkActions”: Number of requests to accept before forwarding them to the cluster.
  - “bulkSize”: Size of data to build up before forwarding requests to cluster.
  - “bulkCloseTimeout”: At the end of the run, we wait for the bulk processor to report on its success or failure with all update requests. This setting controls how long we wait.
  - “bulkCloseTimeUnit”: Time Units for bulkCloseTimeout, e.g., SECONDS, MINUTES.
  - “flushIntervalSeconds”: Max interval to wait between pushing requests to the cluster.
  - “concurrentBulkActionRequests”: Number of bulk requests we send at once.

Migration Groups

Note

Index names and types are examples only

```
migrationGroups:
  - groupName: "my-group-1"
    input:
      indices: ["index1", "index2"]
      types: ["document", "error_message"]
      filter: {exists: {field: "creation_date"}}
    migrations:
      - name: "TruncateUsernames"
      - name: "RemoveCreationDateField"
  - groupName: "my-group-2"
    migrations:
      - name: "LowercaseBodyField"
```

- The “migrationGroups” key maps to a list of ‘migrationGroup” objects.
- Migration Group

    - “groupName’: User-defined name of the migration group, used in logging output.
- Input: expected input settings are generally defined in migrations and merged together by the application. These options should be used as overrides when necessary, and tested with dry-runs before use.
    - “indices”: A list of indices to run migrations against; overrides migration defaults.
    - “types”: A list of types to target a migration against; overrides migration defaults.
    - “filter”: An Elasticsearch query object that will be used as a filter.
    - Elasticsearch queries are expressible in JSON, and YAML can be converted to JSON, so YAML syntax is acceptable here.
- Migration
    - “name”: The name of the Migration must be the same as the "shortName" field of the Migration instance. Use the `--list` flag to see available Migrations.

Usage

Install

Install the Minigator as close to the cluster as possible, meaning in the same local network as the elasticsearch cluster. The thing we want to avoid is having system administrators try to use an "ssh tunnel" to expose the elasticsearch cluster to the Minigator tool. The Minigator streams events over the network, so a fast connection to the cluster is important. Running this on an MDS node is ideal if there are enough free system resources to do so.

You can use java -jar on the Minigator jar-file to execute. Start with java -jar minigator-1.0.jar -h to see a help menu.

Dry Run

Do a Minigator dry-run. This will attempt to run the full suite of migrations and log the IDs and events that are malformed, giving you a chance to fix your data or request migration modifications before running the tool.

```
java -jar minigator-1.0.jar --verbose --es-input
'host=localhost' --dry-run
```

Snapshot

Take a cluster snapshot using whatever means are appropriate for your environment. See Elasticsearch's Snapshot Module documentation if you need to create your own solution.

Upgrade MDS

It will generally be better to upgrade to the new version before migrating. That way, the correct mappings will be in place for new data structures in documents. In specific cases, we may recommend a different migration order.

Run the Minigator

```
java -jar minigator-1.0.jar --verbose --es-input
'host=localhost' --es-output 'host=localhost'
```

Application Monitoring

Logging

The Forcepoint Behavioral Analytics comprehensive logging system provides an in-depth glimpse into how each service is performing. This gives a better sense of specific backend operation, as well as a starting point for tracing bugs when a certain function does not execute as expected. Refer to the below table to access the proper log file for the error you are experiencing:

File Path	Component	Error Type
/var/lib/<service-name>/{conf/logs}/reveal-ui/reveal-app.log	UI	UI HTTP requests
/var/log/ro-mds/mds-server-initd.log	MDS	Startup information written by /etc/init.d/mds-server init script. Look here and in mds-server.out if MDS fails to start.
/var/log/mds-server/mds-server.out	MDS	Significant logging for everything MDS
/var/log/mds-server/mds-server-2016-01-27.out.gz	MDS	Rotated MDS log. How many stored is configurable in mds-server-config.yml
/var/log/mds-server/request.log	MDS	Logs HTTP requests to the MDS API
/var/log/ro-ui/audit.log	UI	This is an audit log used for things like reviewing events.
/var/log/reveal-ui/reveal-ui-init.out	UI	App startup information. Look here if the UI fails to start.
/var/log/ro-qw/reveal-qw-server.out	Streaming Ingest	All things Queue Worker.

File Path	Component	Error Type
/var/log/reveal-api-server/reveal-api-server.out	Streaming Ingest	All things Public API.
/var/log/reveal-conversion-server/reveal-conversion-server.out	Streaming Ingest	All things Conversion Service.

Example File

Error type: Login failure

Log details: All request defaults. The user can be found in req-headers > cookie.

You can also see the user login at the /session/new URL, the page they requested it from (referrer), the email of the user and in the response (res-headers) you can see them being bounced to the /unauthorized page (location).

LogFile:

```
{"name":"express-requests","hostname":"ip-10-10-10-
51.ec2.internal","pid":8275,"level":30,"remote-
address":"127.0.0.1","ip":"127.0.0.1","method":"POST","url":
"/session/new","referer":"http://localhost:4200/
login","user-
agent":{"family":"Chrome","major":"54","minor":"0","patch":"
2840","device":{"family":"Other"},"os":{"family":"Mac OS
X","major":"10","minor":"10","patch":"5"}},"body":{"email":"
ken@redowlanalytics.com","password":"UNDISCLOSED"},"short-
body":"{ email: 'ken@redowl","http-version":"1.1","response-
time":18,"status-code":302,"req-
headers":{"cookie":"connect.sid=s%3AptmZiwSmrfNUPqr5U8NrCFhc
9FGbPIIj.FnQ6U5JsE2%2Fai6BW6PWHSKNMr5vGiTcvPK08e5pNoi8","acc
ept-language":"en-US,en;q=0.8","accept-encoding":"gzip,
deflate, br","referer":"http://localhost:4200/
login","content-type":"application/x-www-form-urlencoded;
charset=UTF-8","user-agent":"Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/54.0.2840.98 Safari/537.36","x-requested-
with":"XMLHttpRequest","origin":"http://
localhost:4200","accept":"*/*","content-
length":"41","connection":"close","host":"localhost:4200"},"
res-headers":{"x-powered-by":"Express","location":"/
unauthorized","vary":"Accept, Accept-Encoding","content-
type":"text/plain; charset=utf-8","content-
length":"47"},"req":{"method":"POST","url":"/session/
new","headers":"[Circular]"},"res":{"statusCode":302,"header
":"HTTP/1.1 302 Moved Temporarily\r\nX-Powered-By:
Express\r\nlocation: /unauthorized\r\nVary: Accept, Accept-
Encoding\r\ncontent-type: text/plain; charset=utf-
8\r\ncontent-length: 47\r\nDate: Mon, 05 Dec 2016 21:20:28
GMT\r\nConnection: close\r\n\r\n"},"incoming":"<--
","msg":"127.0.0.1 <-- POST /session/new HTTP/1.1 302 47
http://localhost:4200/login Chrome 54.0 Mac OS X 10.10.5 18
ms","time":"2016-12-05T21:20:28.288Z","v":0}
```

Graphite and Grafana

Once the application is up and running as desired, it is important to monitor activity to check performance and potential scaling needs. Grafana and Graphite help us gather and visualize the health of various Forcepoint Behavioral Analytics components.

Access Grafana at http://localhost:3000/. Default login/password is admin/admin

Grafana Dashboards

Grafana is a frontend for Graphite which allows for customizable dashboard visualizations and alerting. Each monitored component in Forcepoint Behavioral Analytics has its own Grafana dashboard. The following dashboards are shipped with Forcepoint Behavioral Analytics:

- Infrastructure - key hardware metrics
- MDS - data query performance
- Public API - incoming ingest event stats
- Conversion Service - data and file (natives and attachments) stats
- Queue Worker - ingest processing statistics by type (deduplication, ER, labeling, etc)
- Alerting
  - Process uptime and disk thresholds can be set and alert via PagerDuty, Email, or Slack

Additionally, to supply data to these stores we have new components running on all/ most hosts within the deployment:

- Elastic Beats - gathers logfiles and forwards them to the Logging ELK
- CollectD - gathers metrics from the underlying host/OS and certain third party processes and forwards them to Graphite

Metrics from our Dropwizard-based applications (e.g., MDS, Queue Worker, Public API, Conversion Service, etc) are sent directly to Graphite from the application itself. With the Forcepoint Behavioral Analytics v2.50 release, we have detailed metrics from the following processes:

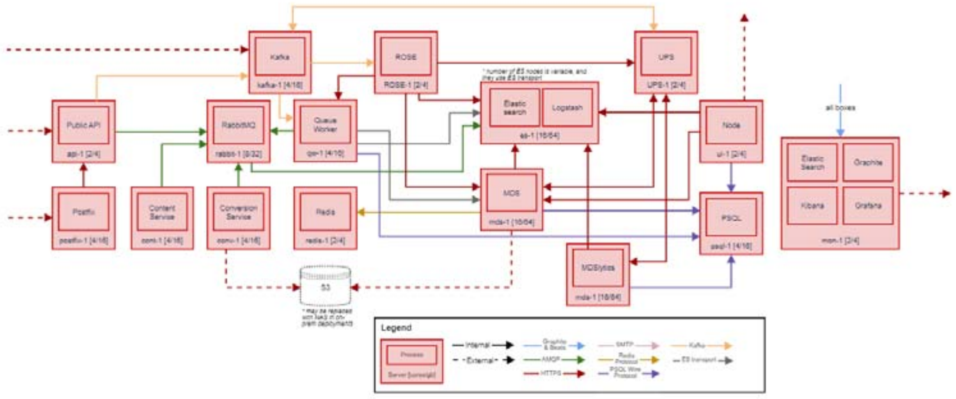
- MDS
- Public API
- Conversion Service
- Queue Worker
- UI Server
- Elasticsearch
- Postgres

- RabbitMQ?

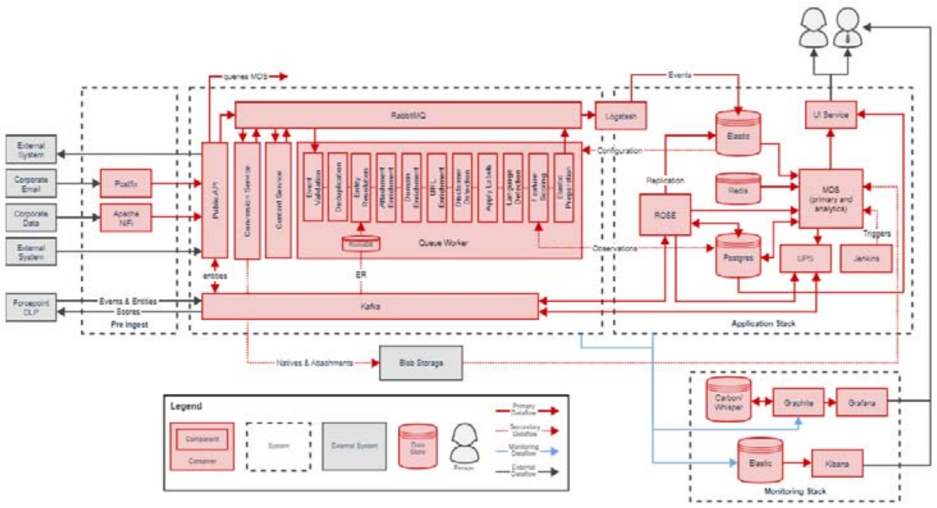
Appendix

Expanded Forcepoint Behavioral Analytics Architecture

3.3 Physical Architecture



3.3 Component Architecture



User Interface - AppConfig

The Forcepoint Behavioral Analytics UI is configurable through the AppConfig API. AppConfig is simply a RESTful endpoint in MDS that gets/sets JSON-based configuration objects. Minor alterations can have considerable impact on analytic results, so take caution when editing. The following attribute tables provide a comprehensive insight into how Forcepoint Behavioral Analytics can be fine-tuned for almost any use case:

event_feature

Variable Name	Description	Example Values
standard_deviation_aggregate_parameter	Used on the Analytic Dashboard when aggregating by standard deviation rarity.	1.5
event_feature_weight_lower_bound	On the Models page, when setting Feature weight, -5 is the default lower bound. This can be configured if necessary.	-5
event_feature_weight_upper_bound	On the Models page, when setting Feature weight, 5 is the default upper bound. This can be configured if necessary.	5

analytics

Variable Name	Description	Default Values
outlier_probability_lambda	A significance coefficient used by the LoOP calculator.	2.0
outlier_probability_neighbors	Used by the LoOP calculator to determine how far back to look (in a number of defined time intervals) to establish the historical baseline.	10
probability_max	There are issues with probabilities of 1 (namely, log(1-1) = -inf). This represents the maximum allowed value for probabilities. The "D" relates to double precision and the use of decimals to be extra precise.	0.99D
overall_average_precision	This defines the level of precision for dividing when calculating the overall average of some aggregates.	10
rank_calculator_neighbors	Determines the historical window (in number of defined time intervals) used when calculating ranks on the Analytic Dashboard.	10

Variable Name	Description	Default Values
enable_entitlement	This variable affects the use of entitlements when calculating entity risk scores on the Analytic Dashboard. The risk scores have to be calculated for each entitlement (because entitlements hide events) which increases the number of computations that need to be done. By default we want to **disable** entitlements to reduce the performance impact.	true
score_comparison_intervals	How many intervals to show score comparisons on the Analytic Dashboard's Score Comparison visualization (default one week's worth of days). Value is in days.	7

score_comparison_interval_length	Size of score comparison intervals on the Analytic Dashboard (default one day) in a "parsable duration" using this pattern: PnDTnHnMn.nS (
	nD = number of days,
	nH = number of hours,
	nM = number of minutes,
	n.nS = number of seconds, the decimal point may be either a dot or a comma.
	T = must be used before the part consisting of nH, nM, n.nS)
		P1D

score_comparison_categories	Use these Entity Attributes as the score comparison categories on the Analytic Dashboard to group entities (default location and department).	Location, Department
score_comparison_buckets	Number of buckets used to group score comparisons on the Analytic Dashboard's Score Comparison visualization (default quintiles, example: 0 - 19, 20 - 39, 40 - 59, 60 - 79, and 80 - 100).	5

Variable Name	Description	Default Values
recent_intervals	How many intervals to show for recent intervals on the Analytic Dashboard (default one week's worth of days).	7
recent_interval_size	Size of each recent interval (default one day). Alternate options include: WEEK, DAY, HOUR, MINUTE, SECOND, but DAY, WEEK, or HOUR are suggested.	"DAY"
recent_interval_spacing	Spacing of each recent interval (default one hour). Alternate options include: WEEK, DAY, HOUR, MINUTE, SECOND, but DAY, WEEK, or HOUR are suggested.	"HOUR"
minimum_cardinality_precision	Minimum precision threshold for cardinality aggregations on the Analytic Dashboard. L represents the java data type and is the only data type supported here. When changing this value, you do not specify the datatype. The change would look like: {"minimum_cardinality_precision":123456789}	20L
time_between_precision_threshold_updates	AnalyticDataDAO periodically pulls the cardinality precision threshold from AppConfig, as a means to prevent hitting ES every time a request is made. This determines how frequently that trip to ES takes place (it already pulls the value by default on MDS startup). Default is in ms.	86400000
required_hours_of_metadata	The number of hours of cached entries required in order to not display an error to the end user. Example, assume a client updates their Analytic Dashboard every hour. If we go 2, 3, or 4 hours without seeing an update, we should notify the client that their data hasn't been cached.	4

Variable Name	Description	Default Values
analytic_scenarios	List of Scenario root IDs representing the Scenarios we want to show on the Analytic Dashboard. Default of empty means all Scenarios will be used.	[]
millis_between_metadata_last_used_update	There is a value to indicate when a cache entry was last read. We used to update it after each update, but that proved costly. We not only update the value once an hour (in milliseconds) and this should almost never be changed.	3600000
analytic_endtime_override	Analytic Dashboard's default end time. Zero (0) is the default value and means, "Use the present time / the last timestamp in eventdata". Otherwise this configuration takes a timestamp (in milliseconds) to use as the endtime of the Analytic Dashboard. f the timestamp is invalid (< 0 or > present), MDS will error upon trying to load the Analytic dashboard.	0

dst

Variable Name	Description	Default Values
dataset_timezone_id	This variable tells MDS which timezone to use as default when interpreting dates and timestamps in the app. Timezone ID as indicated by the standard list of timezones seen here: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones Note: it is required to set both the timezone ID and the timezone offset (below) because some code uses one and some code uses the other.	America/New_York
dataset_timezone_offset	Timezone offset as indicated by the number of hours from GMT.	-500
event_types_with_context	This configuration relates to the "view more" button within events and denotes which mode types should show more events ("view more chats"). This is not related to the Show Context or Show Context Configurations feature and page respectively. It should be by mode.	["chat"]
date_picker_start_in_days_from_end		
explore_page	This setting denotes what the Explore page's end date is. This setting does not control datepickers on other pages. The value is in days.	1
event_attribute_rendering		
disable_all	This designates whether to show event attributes in all Event Viewers or not. This is an all or nothing designation.	false

Variable Name	Description	Default Values
time_format	This designates whether to use 12 hour or 24 hour time for the datepicker time format. Available options include, but are not limited to 12 hour and 24 hour. https://momentjs.com/docs/#/parsing/string-format/	h:mm
date_format	Used to configure how users will see dates throughout the application. Other examples are: YYYY/MM/DD and DD/MM/YYYY	MM/DD/YYYY

event_references_context

Variable Name	Description	Example Values
alert		
time_before_event	Specifies the number of milliseconds before the events to include. The offset is converted to an absolute date and time relative to the earliest referenced event. The show context RQL filters out events prior to that date/time.	432000000
time_after_event	Specifies the number of milliseconds after the events to include. The offset is converted to an absolute date and time relative to the latest referenced event. The show context RQL filters out events after that date/ time.	86400000
modes	Sets the default mode filter for results. The show context RQL will limit results to any of the specified modes.	email, chat
entity_fields_from_referenced_events	Contains one or more Entity roles from the old data model e.g., sender_not_analyzed. The Entities in these roles in the referenced events are added to the show context RQL.	traderId_not_analyzed
Entity_role_ids_from_referenced_ events	Contains one or more Entity roles from the new data model e.g., sender_rMajHapDFh. The Entities in these roles in the referenced events are added to the show context RQL.	trader_o8mnYeQtUm
additional_rql_context	Contains any extra RQL to include. The show context RQL will include the specified RQL.	""

event_type_configs

Variable Name	Description	Example Values
voice_event_types	These are the event types that will display a transcript and the option to play an audio clip if it exists. Alternate values that can be used are event modes.	voice

review_status

Variable Name	Description	Example Values
closed_status	This is the Review Status ID for the review status: Closed, which is be used in every environment. It is base64 encoded.	Y2xvc2VkOiByZXNvbHZlZA==

event_context

All of these Show Context configurations can be changed in the UI and should only be changed there.

Variable Name	Description	Example Values
mode_defaults		
time_before_event_ms		43200000
time_after_event_ms		43200000
modes_in_context		[]
entity_roles_on_original_event		[]
entity_role_ids_on_original_event		[]
rql_filter		[]
configurations		null

mds

The UI now supports the ability to define the order in which entity roles appear on the Event Viewer. Previously you could only hide or override the name of these roles. Roles that are present in the event and listed in the sort order configuration will be displayed first, in the desired order. Remaining roles are to be displayed in alphabetical order. If multiple roles on an event are defined with the same order, the output cannot be guaranteed.

Variable Name	Description	Example Values
entity_roles_config		
order		1
hidden		true
displayName	Display name override	“BCC”

Data Format

Events

Forcepoint Behavioral Analytics uses event data to build analytic reports. They are stored in Elasticsearch at <es>/eventdata/<mode> (or one of the time-sliced indices aliased to eventdata).

External clients typically provide events to Forcepoint Behavioral Analytics with some subset of the following core event components:

- timestamp
- mode (e.g., email, phone, trade, vpn log, badge swipe)
- entities involved (e.g., people, files, network device, phone number, bank account)
- unstructured content (e.g., body of an email, transcript of a phone call)
- structured content as event attributes (e.g., email headers, financial transaction amounts, call duration, inbound/outbound bytes)
- attachments
- event references - direct links to other events in the system
- transcription metadata (specific to audio events where we have a transcript of the given audio file)

Forcepoint Behavioral Analytics adds a variety of metadata to these events via streaming ingest, batch analytic jobs, and/or annotations directly from the UI:

- review status
- labels
- extracted features and their associated probabilities
- denormalization of the provided entities (e.g., "Jane Foo" when given her phone number "867-5309")
- storage info for native event copies
- storage info for attachment binaries

Modes

Forcepoint Behavioral Analytics uses Modes to classify variations of monitored communication and activity events. The Mode list includes but is not limited to:

- Alert
- Authentication
- Chat
- Data-movement
- Email
- Physical
- Process
- Trade
- Voice
- Web
- Web-search

(public JSON format)

Entities

Entities are the core of an event (e.g., people, files, network devices, phone numbers, bank accounts). Forcepoint Behavioral Analytics prioritizes Entities based on the impact of the events they are involved in. Events store "raw" entity identifiers, provided by the data source, as well as "resolved" Entities, which represents a core Entity (if any) that the raw identifier has been linked to (e.g., the person "Jane Foo"). Entities are stored in Elasticsearch at <es>/refdata/actor.

Entities have the following components:

- Resolved Entity Name - e.g., "Jane Foo"
- Identifier - e.g., the email address "jfoo@gmail.com"
- Aliases e.g., "jfoo@gmail.com", "456-1234", "workstation-jfoo"
- Attributes - structured metadata about the Entity (similar to event attributes)

```
{
  // entities live in the refdata index
  "_index": "refdata",

  // document type. this is 'actor', should be 'entity'
  "_type": "actor",

  // identifier. should be a UUID, is currently a string.
  "_id": "Laura Stahl",
  "_version": 2,
  "found": true,
  "_source": {

    // actor attributes: a detailed description is outside
the scope of this page,
    // see: https://redowl.atlassian.net/wiki/display/REV/
Searching+events+using+actor+attributes:+Technical+spec
    "attributes": {
      "boolean": [
        {
          "id": "204863ee-ff93-4132-a27d-22140d94a6cd",
          "name": "Dirtbag",
          "value": true,
          "attributeType": "BOOLEAN"
        }
      ]
    },

    // identifier. this is probably unnecessary, and should
be a UUID if it is necessary.
    "actorId": "Laura Stahl",

    // name to display in Reveal
    "displayName": "Laura Stahl",

    // list of aliases to use for entity disambiguation
purposes.
    "alsoKnownAs": [
      "7102745960",
      "lauras@thp.com",
      "215.206.100.222",
      "lstahl"
    ],
    "oldAttributes": null
  }
}
```

Dataset Metadata

Forcepoint Behavioral Analytics stores a variety of metadata in <es>/
dataset_metadata/<object>, where the "object" includes the following:

- application configuration
- lexicons
- feature configurations
- analytic "scenarios"
- review status
- trained classifiers
- entity resolution keys

Generic Entity Roles

Entity Roles classify the Entity involved in an event, and help us define information movement, like origin and destination, and physical classification of an entity, like person, printer, etc.

- Content: sender, recipient
- Trade: portfolio manager, trader, security
- Digital: modifier, origin, destination, origin device, destination device

Content Model Example:

```
{
   "_id": "2797327a-27b9-45fd-a905-dd8fcdd8a21f"
   "_source":
      {
          "sender":"paul@redowlanalytics.com"

"recipients":["mikeg@redowlanalytics.com","det@redowlanalyti
cs.com"],
          "bcc":["paul@redowlanalytics.com"]
          "subject":"OG"
          ...
      }
}
```

These pages routinely provide summary statistics and rankings for the fields mentioned above:

- Explore: Top Entities, Activity Over Time (by entity), Comparative Timelines, Chat Matrix, Top Trades
- Scoring: Ranked entities (primary role) broken out by ranked relationships (secondary role)
- Analytics Setup Models: Primary and secondary role selections
- Behaviors: Reports and Profiles are entity rankings, where the user configures the "Entity Type" (these are entity roles)

- Entities: All entities tracked within a Forcepoint Behavioral Analytics deployment can be found here. Filter out by specifying attributes, dates, and / or searching for specific entity names
- Dashboard: Various views into the previous items
- Settings Entity Resolution: All of these fields have "resolved" variants that allow you to link the same entity across multiple modes
- Settings Entitlements: Our entitlements are currently based on the values available in the entity fields
- RQL and QDSL: Several components of the Forcepoint Behavioral Analytics search syntax rely on these fields
- UTA: Same considerations as the combination of Behaviors and Scoring

Developer Tips and Tricks

Logstash

Once data is output from the Queue Worker, Logstash grabs event, Entity, and metadata from RabbitMQ and pushes it to Elasticsearch. Logstash is the weakest link. While events entering "elasticsearch.event.queue" from Logstash are persistent, there is no guarantee that Logstash will put those events in Elasticsearch.

- Events are acknowledged as soon as Logstash is able to fetch them and not on successful Elasticsearch writes. If the Elasticsearch write fails, due to an incorrect Logstash configuration for example, this may cause Events to be dropped where messages are fetched from the queue.

Start the Services

- sudo service mds-server start (if this isn't already running)
- sudo service reveal-api-server start
- sudo service reveal-conversion-server start
- sudo service reveal-qw-server start
- sudo service logstash start

Activate services

- sudo chkconfig logstash on
- sudo chkconfig ro-api on
- sudo chkconfig ro-qw on
- sudo chkconfig ro-conv on

Elasticsearch

Elasticsearch is an efficiently searchable, enterprise-grade search engine that stores all Forcepoint Behavioral Analytics events. Data in Elasticsearch is queried through MDS.

Elasticsearch Best Practice

Cluster Checklist

1. Ensure that max_file_descriptors is at least 64,000 on the underlying OS.
2. Use machines with 64GB of RAM and 2-8 cores.
3. Allocate no more than 32GB to the java heap, ever (The JVM cannot use compressed object pointers and everything will be slower above 32GB).
4. Allocated 50% of total RAM to MDS/Elasticsearch as best practice. Leaving enough memory free for the OS is critical because Lucene will use a lot of off-heap memory.
5. Use Concurrent Mark and Sweep GC.
6. Prefer SSDs; they are far faster than any other local storage. Disk usage should be under 50% of disk available, as write-heavy operations will spike disk usage. If using SSDs, ensure that the I/O scheduler is either "deadline" or "noop", as the *nix default of "cfq" is optimized for spinning media.
7. Ensure that the JVM running Elasticsearch isn't allowing swap to disk. The setting in Elasticsearch to ensure that the JVM will not swap to disk is bootstrap.mlockall=true.

Example of the JVM settings derived from the above tips:

ansible sets the following:

mds_java_opts:

```
- "-Xms{{ mds_java_opts_xms }}"
- "-Xmx{{ mds_java_opts_xmx }}"
- "-XX:+HeapDumpOnOutOfMemoryError"
- "-XX:+UseConcMarkSweepGC"
- "-XX:HeapDumpPath={{ mds_heapdump_path }}/"
- "-XX:+AlwaysPreTouch"
- "-XX:+UseCompressedOops"
- "-XX:+UnlockDiagnosticVMOptions"
- "-XX:+PrintCompressedOopsMode"
- "-XX:+PrintFlagsFinal"
```

Use the follow to check max file descriptors:

```
cat /proc/sys/fs/file-max
```

Elasticsearch Tuning for SSD

By default, Elasticsearch limits the allowed aggregate bytes written across all merges to 20 MB/sec. For spinning disks, this ensures that merging will not saturate the typical drive's IO capacity, allowing concurrent searching to perform well. If you are not searching during your indexing, search performance is less important than indexing throughput. If your index is on SSDs, you should disable merge throttling entirely by setting index.store.throttle.type to none; see store for details.

Exclude Unnecessary Field Data

It is best practice to exclude unnecessary field data, especially for large attachment bodies. OutOfMemory errors in Elasticsearch queries are caused by the inclusion of large fields, which happens by default. Do not forget that you are aggregating events themselves, not just their counts.

When processing results of queries, instead of deserializing the full JSON document, consider parsing only the necessary field. If using field selection, full object serialization is not an option. Anytime you are serializing the results of a query, ask yourself if you really need the full object (which can be large in many cases).

Query Performance

The following best practices elaborate on how Elasticsearch queries can be performant across a variety of communications datasets.

Prefer iterative queries over deeply nested facets

- As a level-up over faceting, Elasticsearch 1.x introduced aggregations. Do not underestimate the amount of memory usage that nested aggregations require. Nesting performs worse than iterative queries in many cases. As Elasticsearch improves facet optimization, we expect this to change, but for now, it is a real consideration.

Be aware of field cardinality

- Nested sender/recipient aggregations blow out memory in high cardinality fields. For example, doing graph style aggregations where you want to select senders, and then recipients for each sender, can take a tremendous amount of memory. Beware of aggregations using fields with high cardinality.

Make sure the query is efficient

- Ensure that all parts of a query are being actively used and remove unused aggregations.

Restarting and Indexing

If you are spending lots of time reindexing or watching shards balance on a cluster with lots of data, the following best practices may be helpful.

Turn off refreshes during indexing

- An Elasticsearch refresh is a low-cost operation that writes the in-memory buffer to disk in order to make it available for search. By default, this happens once per second. During a large import, you may want to turn this off in order to speed up the import.

Disable replication during indexing

- If you have primary and replica shards, each shard will have to analyze a document as it's indexed, so every analysis will be performed twice. To analyze each field only once, turn replication off while you index. When you turn replication on, the replica will rely on the primary shard's analysis and simply copy from the primary shard.

Flush the indices before restarting the cluster

- The heavier version of a refresh is a "flush," which not only writes information to disk but also does a "full commit" of the Lucene segment. These are scheduled to run every thirty minutes. This means that your node may have to replay thirty minutes' worth of activity before completing shard recovery. It's better to flush beforehand because longer shard recovery times tend to set off elaborate automated re-balancings.

Backups / Snapshots

Create Repo

```
curl -XPUT 'http://localhost:9200/_snapshot/
default_s3_repository' -d '{
  "type":"s3",
  "settings":{

"bucket":"redowl.nextera.mds.elasticsearch.snapshot.orange",
    "server_side_encryption":true
  }
}'
```

Create Snapshot

```
curl -XPUT "http://localhost:9200/_snapshot/
default_s3_repository/snapshot_$(date +%Y%m%d%H%M%S)"
```

Helpful Information

Vagrant (used for internal testing) runs virtual machines. In this case it's running a vm which is running graphite. There are two ways to talk to graphite. One is through its ssl port 443. The other is its UDP port, 2003.

Seeing Grafana data

To output data use the udp port:

```
https://localhost:443/
```

Once data is input (see below), you should be able to see a file structure, which under "Metrics" contains whatever namespace used. Clicking on this, adds to the graph.

1. Be aware of the scale of the x axis, configurable using clock.
2. The graph refresh button doesn't always work, but changing the line mode (Graph options ? Line Mode), is effective at refreshing the graph.
3. Try the following if your server crashes mid-process
   a. vagrant destroy (warning: will cause data loss)
   b. vagrant up

Grafana Data Input

To import data, we use this pipe command:

```
echo "foo.bar 5 `date +%s`" | nc localhost 2003
```