Constellation Analytics Platform Book

From Haystax Analytics Developer Wiki

Contents

Introduction

What is the Constellation Analytics Platform? It is a software product developed by Haystax Technology to help solve problems of prioritization. We consider a model first to address problems we can solve in many domains. We developed a platform to allow analytic processing using Bayesian statistical modeling, data ingest from a variety of sources to provide evidence, and interfaces for users to input data, catalog assets (important things), and view results to use the system in a real-time manner.

The Constellation Analytics Platform is a software system providing users tools for prioritizing threats against things they value. This is the simplest way to describe what the entire system does, as it ranges from data collection, model evaluation to assign threat and risk scores, asset cataloging, analysis, and results displays. It is a cloud-based offering (with on-premisis options) supporting web- and mobile- based applications. The platform provides the base, applications, data, and tools for users to be able to solves many issues in their particular domains ranging from public safety to insider threat.

The platform consists of several components. These include data storage, data processing and model evaluation, API for results and data access, and a user interface providing ways for users to add, analyze, and view asset information, threat information, and model results. These components may be used and installed in different configurations depending on the use case of the client. The platform itself was developed to be cloud-based, using cloud data storage, scalable clusters for analytic processing, cloud third party authentication and authorization, and web-based interfaces along with a mobile component. The platform also uses many open-source libraries and standard interfaces such as RESTful APIs at many layers. The entire platform can be installed on one laptop, or scaled up to dozens of machines with redundant sharded data storage and processing clusters.

Model evaluation (the back-end analytics)

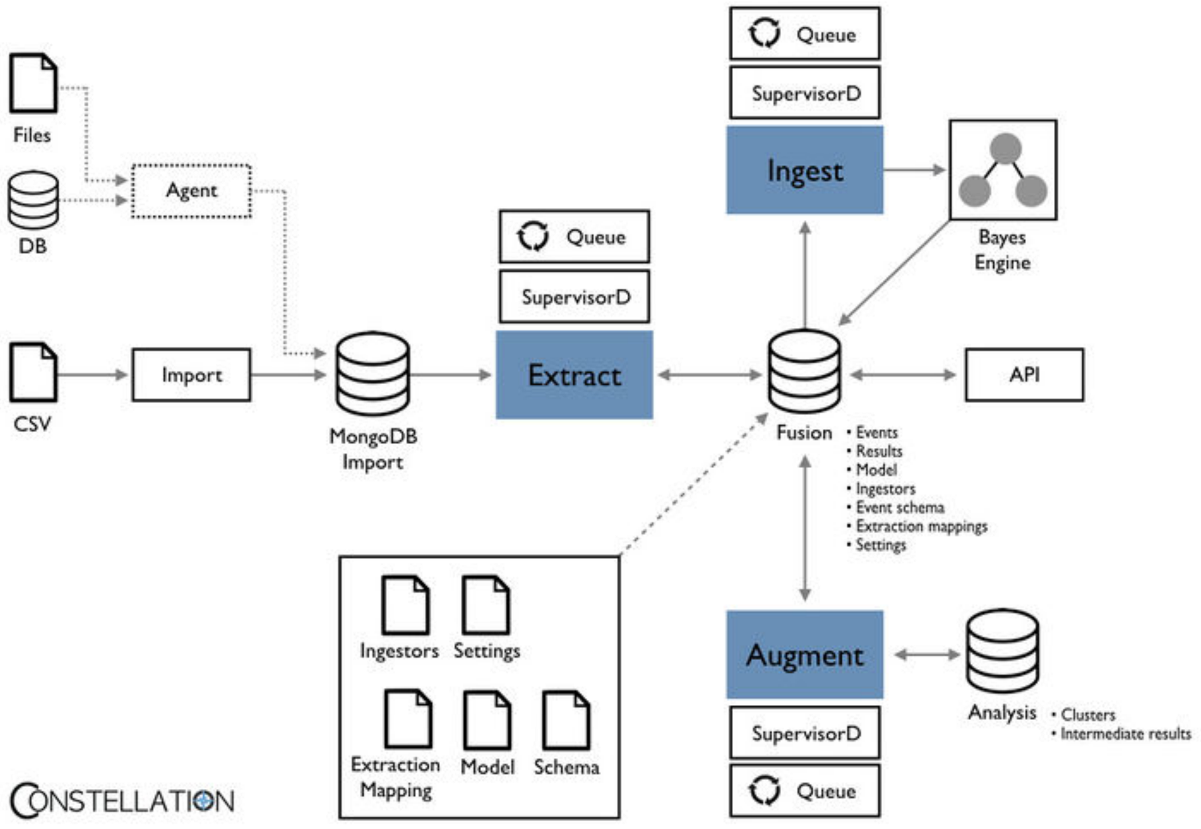
Constellation Fusion models represent a way to reason about a problem. For example, the Carbon insider threat model represents how an expert decides if a person is trustworthy. This is a very high-level belief concept - and when you think about what that means you have a mental model about how you break that down into sub-concepts. At some point if you go down far enough and break concepts down, you should reach a point where actual data and observations can be directly applied to a model node. For the model to give an accurate answer, findings which are observed in data, such as person tested positive for drug use, or computed findings, such as person entered the building at an unusual time, must be be applied to the model.

Fusion apparatus

Fusion is the name of the system that extracts and transforms various sources of data into observed findings, augments findings by deriving or computing new findings, ingests findings into the model and gives us the answer to a question the model was built to answer.

While the deployment can change, there are some basic steps that the system uses to get data from raw form into actual model results of belief. The data is read from raw files and translated to be stored as an event on an object schema. Think of some data exfiltration event attached to an actual person, formatted correctly to match our object definition. The data can then be augmented by performing calculations to determine in an event is anomalous, or other uses of machine learning and NLP to process the event data in preparation for the model mapping. The data is then Ingested into the model, by mapping object event attributes onto nodes in a Bayesian model. The model is then run, performing the mathematical calculations to compute a belief in important nodes such as Trustworthiness and Issues of concern, which can help prioritize assets (people) for investigation. The results are then stored into a data store for use by the user interface or other systems through the use of an API.

The Fusion steps are made up of individuation micro services implemented in Python. These respond to task requests put on queues that are monitored to know what job to work on next.



Before we dive into the steps of processing data through the system, let's cover the supporting artifacts that drive the data to the model for calculation.

Model spec

The model is encoded as JSON an describes all of the nodes and the structure of the model. This is translated into an actual Bayes net for use in the Bayes calculation engine.

Data transform files

Data files are described as mapping files that define a transform path from data attribute in a file to an event attribute on an object. For example, a particular CSV file column maps to a particular event schema key. These can include conditions like regular expression matching to filter out data values based on conditions.

Events schema

First, we look at model nodes and identify events that could be applied as findings. For example, Carbon model has PoorWorker node. What event in a person’s life could be a positive indicator that the person is a poor worker? Unsatisfactory performance rating could be one such indicator. So we create an UnsatisfactoryPerformanceRating event and since this event occurs at a point in time, we mark it as a point event. If an event spans a period of time and has a start and end date, the event is durative. There might also be event types that are never applied as findings, but instead are used to calculate or derive computed findings. For example, to determine if a person entered the building at an unusual time (UnusualAccess event), we need to look at all access events for the person. All event definitions are specified in constellation.carbon package under schema/events/events.csv CSV is converted into JSON schema, which is used to transform and validate source data when it is extracted into format that is understood by ingest and augment steps.

Event-Node mapping (Ingestors)

Second, after the events schema is completed, event types are mapped to model nodes and decay/growth half life along with the strength and the positive or negative polarity of the finding are specified. For example, Unemployment event is a strong negative indicator for CommittedToCareer node with growth half life, or how quickly the event grows in relevance over time, of 120 days and decay half life, or how long it takes for event to lose its relevance, equal to the duration of Unemployment event. All ingestor definitions are specified in constellation.carbon package under ingestor/ingester.csv CSV is used to generate model ingestors, which are responsible for ingesting the findings into the model in Bayes Engine.

Data

TODO

Extract

The extraction process pulls data out of import data and puts events on objects. It loads data, and then transforms it into a format needed for the event schema.

Load

The process of loading raw source data into the fusion platform is called **load**. We think of raw source data as comma-separated text files (think spreadsheets), structured key/value pair files (JSON), or potentially external databases with data. The raw data is quickly imported into an intermediate database to support the next step, Extraction.

A **data connector** is a way of connecting to a data source and emitting that data into the imports database. Fusion ships with one data connector called MongoImport.

MongoImport uses the mongoimport[1] (https://docs.mongodb.com/manual/reference/program/mongoimport/) tool. Currently, it supports CSV and JSON files. A corresponding data transform file needs to be created for each data source you load. A **data transform file** is a JSON object that contains logic on how to extract events that conform to the events schema. Once the source data is loaded a message is sent to extractor saying "there is new data to be processed."

Transform

The Transform Executor waits for a message from the importer stating that new data is ready to be processed. Each message contains the collection name in the imports database where new source data was imported. Transform attains the corresponding data transform and starts iterating through the documents in the collection. Through each iteration, transform runs through the events in the data mapping and starts creating event objects based on the source data. The creation of events from source data is referred to as **transform**.

Once an event object is created it is run through the **JSON Schema Object Cleaner**. This does simple data cleanup on the event object. For example, the pid must be a string. If the source data contained an integer as the pid, then it would be converted to a string. Once the event object is cleaned a check is done for duplicates in the events collection. If no duplicate is found then it is run through a validator. This validates the event object against the events schema. A field is added to the event object stating whether it passed validation or not. The event object is then saved to the events collection in the fusion database.

Event Object Transform Example

A document from the AccessData source data stored in the customer_AccessData collection:

```
{
        "_id" : ObjectId("589b40a7e6426baadff595c1"),
        "Employee Identifier" : "M000773",
        "Reader Description" : "BLD1-01 T/S-IP #1",
        "DateTime" : "10/01/2015 08:07:30 AM",
        "Transaction Type" : "Open",
        "Status" : "Active",
        "fusion_waiting" : true,
        "fusion_done" : false,
        "fusion_working" : false
}
```

Access source data's corresponding data transform:

```
{
        "_id" : ObjectId("589b40a3e6426baadff56411"),
        "source_collection_name" : "customer_AccessData",
        "tenant" : "customer_tenant",
        "model_version" : "1.0",
        "mapping" : {
                "pid" : {
                        "field_name" : "Employee Identifier"
                },
                "event" : {
                        "access_point" : {
                                "event_date" : {
                                        "field_name" : "DateTime"
                                },
                                "type" : {
                                        "preset_value" : "AccessPoint"
                                }
                        }
                }
        },
        "model_name" : "carbon"
}
```

Augment

Augmentation process

When data is extracted into the system, it is often beneficial to perform augmentation. Augmentation is a process which creates new events on an object by using analytics over processed and external data. What does this mean? Well, suppose Fusion has extracted hundreds of building access points for a person. Suppose almost all of these access points occur between the workday hours of 9am to 5pm. Now, suppose one of the events occurs at 1am. If Fusion could generate an anomalous event from this, it might be used as evidence in a model. But note this anomalous event is not explicit in the data. Only after reviewing hundreds of access points could it be determined that a particular building access point point is anomalous. The creation of this new event is referred to as **augmentation**. It is also possible to "fill in" missing properties on an event. For the building access point event, suppose it has the properties of *event_datetime* and *weekday*. And perhaps, the data extracted only has *event_datetime* values such as *2/8/2017*. From this date, we'd like to infer the weekday (i.e., Wednesday). This is also considered an augmentation task, even though it's much easier to compute and does not result in a generated event. It only modifies a property of an existing event.

Augmentation Anomaly Detection Example

Instead of writing in the abstract, this section gives an example of one particular augmentation task: anomaly detection. Suppose Fusion begins extracting thousands of access points events for thousands of employees. Now, recall how a model evaluation works. A model evaluation requires an evaluation date and retrieves relevant events on or before the given evaluation date for a given pid (person id). Assuming augmentation has been ran, no access points will make their way into model evaluation (they have to be anomalous to be considered as evidence for the model). In order to calculate the anomalousness of an access point, one way is to consider the time it took place for a given pid. If we consider plotting times of access points on a one dimensional line, we might notice clusters of points that occur. These clusters correspond to major events for a given person (e.g., person arrives at work, leaves work, etc). By identifying clusters, data points outside of these clusters, are defined to be anomalous.

There exists a continually process running known as a **Continuous Cluster Publisher**. It's job is to publish tasks to a queue. In this case, the task would like the following:

```
{
  "end_date_range" : ISODate("2015-12-01T00:00:00Z"),
  "event_type" : "AccessPoint",
  "event_property" : "event_date",
  "eps_units" : "seconds",
  "start_date_range" : ISODate("2015-10-01T00:00:00Z"),
  "security" : {
     "tenant" : "demo",
     "readers" : [ "demo" ],
     "writers" : [ "demo" ]
  },
  "min_samples" : 3,
  "pid" : "M000773",
  "eps" : 3240
}
```

Here, the task is to identify anomalous events for the pid *M000773*. Using an event type of *AccessPoint* and the event_property of *event_date*. The start date and end date range what data points should be considered. Now, upon publication of this task, another continually running process known as a **Continuous Cluster Consumer** will pick this task up and begin evaluating the relevant AccessPoints. However, if an access point has been previously evaluated as being anomalous or not, will not be evaluated again. This avoids using event data after the AccessPoint to inform its anomalous. That is, future data will never inform the anomalousness of an event. If a AccessPoint is identified as anomalous, a new event will be generated and added as a relevant event for ingestion by the model.

Ingest

Ingest applies object events to nodes in the model and evaluated the model to product results.

All valid events for a person that occurred before an evaluation date and only events for which there are ingestors are applied to nodes in a Bayesian model. The Bayes Engine performs calculation for each node and propagates results through the network. This sets belief values on the nodes. The high-level hypothesis node(s) are what ultimately give a final belief in how threatening or risky a thing is given the evidence applied. The output of the model run for a person contains all the nodes in a model with posterior belief. The results along with events applied are stored in the database and can be accessed through the API.

Ingest Example

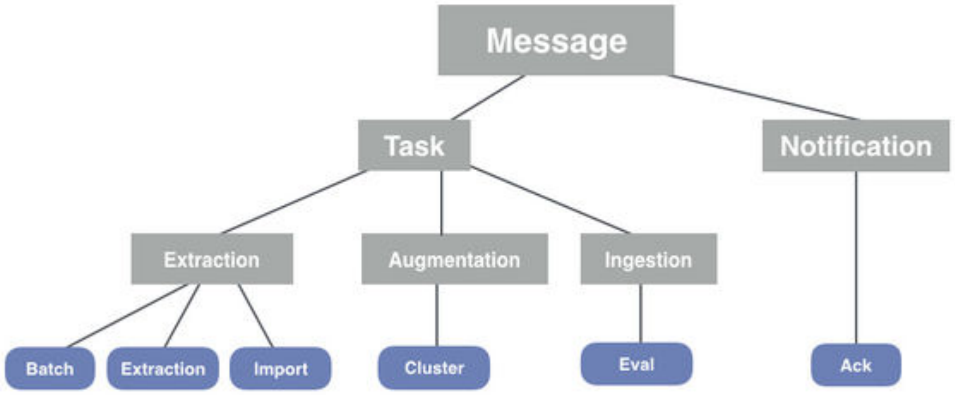
Below is an example `ingestors.csv` file snippet.

```
event,model_node_name,subject_concept_full_name,type,strength,growth_half_life,decay_half_life,is_positive_polarity,max_relev
AccessingConfidentialInformationCase,UnauthInsideITInfAccess_Eff,UnauthorizedInfoAccessAsITInsider,point,strong,,3650,TRUE,1
AdultFinancialExploitationOrAbuseCase,ManipulatesSelfishly_Eff,ManipulatesOthersSelfishly,point,strong,,3650,TRUE,1
AdverseEmploymentTermination,TerminatedUnfavorably_Eff,TerminatedUnfavorablyByEmployer,point,absolute,,1825,TRUE,1
AdverseEmploymentTerminationDisciplinaryAction,DisregardsEmploymentRules_Eff,DisregardsEmploymentRules,point,strong,,1095,TRU
AdverseEmploymentTerminationDisciplinaryAction,TerminatedUnfavorably_Eff,TerminatedUnfavorablyByEmployer,point,absolute,,1825
```

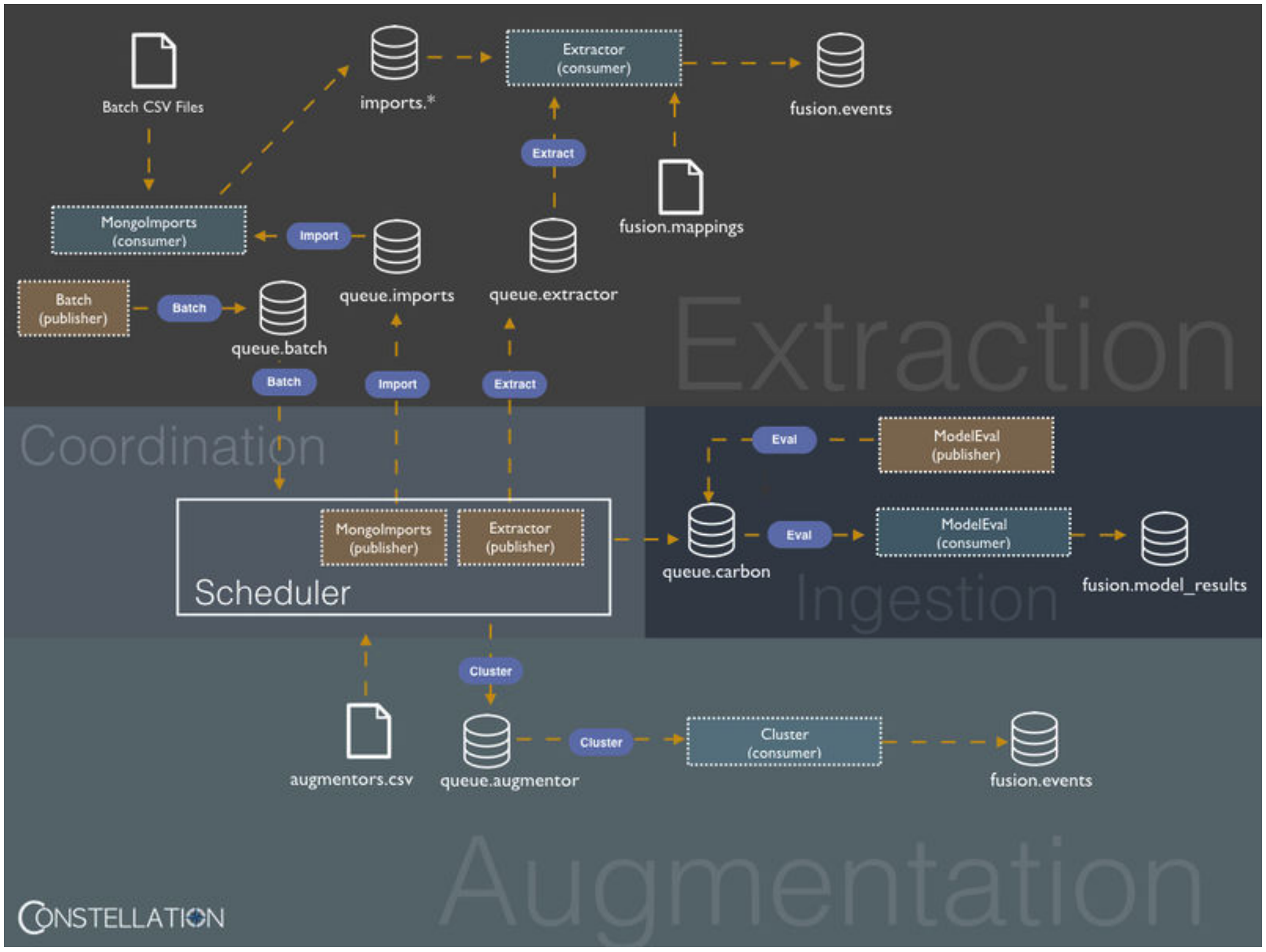
Coordination

Parts of the system which control jobs, tasks, queues, etc.

A fourth component, known as Coordination, works to coordinate the activity of the three main components. The various components of Fusion communicate via message passing. Fusion uses asynchronous message passing. The basic idea is that processes can communicate with each other without having to wait. A classic example is text messaging. With a sender gives a receiver a text message, the receiver is not forced to respond immediately. So what are these messages? The taxonomy below gives the taxonomy of messages used in Fusion:



There are two types of messages: tasks and notifications. A task is something like find anomalies for this person under this given date range. By allowing message passing to occur, this allows Fusion to receive task messages from anywhere (e.g., API call, cronjob). When a task is done, an ack message can be generated. This allows other processes to optionally listen for these messages and then themselves begin some type of work. Messages get stored in things known as queues, which processes can read/write to. This is the solution, to allow manage many async processes. The next diagram gives a detailed view of how the various components communicate with each other:



Model development

TODO

Data presentation (the UI)

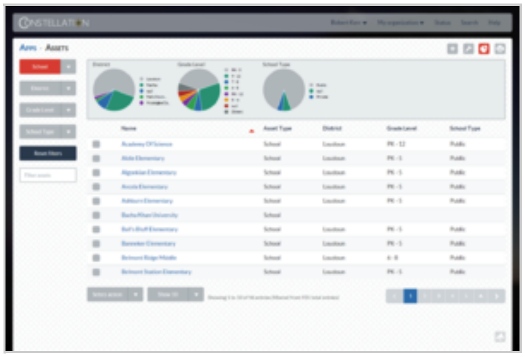
The Constellation web application provides a collection of user focused apps to support public safety and security workflows to view analytical results, track assets, perform assessments, manage incidents, view threat information, and see it all on a map. The interface provides a way to create / edit / read data and results. The platform provides an Application Programming Interface for interactions with the web front end, mobile applications, and third-party integrations. The platform is cloud-based but can be installed on-premisis in a disconnected manner.

Apps

The Constellation user interface provides many apps to assist users in adding data, tracking incidents, and viewing results.

Assets

The assets section is useful for cataloging data about assets important to a client. The types of assets supported currently are schools, places (general buildings), and people. These asset types are driven by a template so we can add any new types of assets we wish (not available to the users, but this is a developer task). The general format of the Assets app is a list screen and a view screen. The list screen presents a data table list of assets with columns of data. There are also filters on the left-side to allow the user to filter on attributes, and a search box to narrow down data table results. On the list screen are also buttons for a mini dashboard (pie charts of counts by filtered attributes), print view, and a button to create a new asset. The data table may be sorted by column headers, and also paged to see more data than the initial ten rows.


Asset list screen with mini dashboard

From the list screen the user may select an asset and go to the view page. The view page has sections for Overview, Details, Photos/Documents, Map, Points of Contact, and Annotations. On this screen the user may edit information about the asset in either the Overview or Details sections. The user may upload associated images and documents. The user may set the location of the asset. The user may add annotations about this asset and choose to make these annotations private to themselves or share with the tenant or data groups.

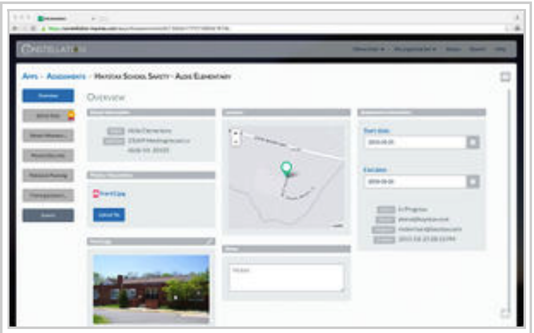
The roles appropriate for this app are tenant_role_asset_get, tenant_role_asset_delete, tenant_role_asset_post, and tenant_role_asset_put.

Assessments

The Assessments App allows users to perform assessments against assets in their catalog. The assets are driven by imported templates which may be customized (by developers or field engineers). The assessments are a list of sections, each of which contain questions about the assets. Templates may be tenant-specific.

The list screen provides a standard data table view of assessments with columns for attributes of those assessments. The columns may be sorted, and the table may be paged to see more content. There are filters on the left side to allow users to filter on important attributes of the assessments, and a text search for finding assessments. There is a mini dashboard showing pie chart breakdowns of the data attributes, and a printed report view. There is also a new rollup report which will show aggregates of assessment questions over multiple assessments.

The user may create a new assessment by clicking on the '+' button. The user will choose an assessment template first, then choose an asset to assess. The user may assign the assessment to a user by entering an email address. The start date and end date should also be entered. When finished the 'Create assessments' button should be pressed. To cancel without creating an assessment, just press the '+' button again to close the create window.

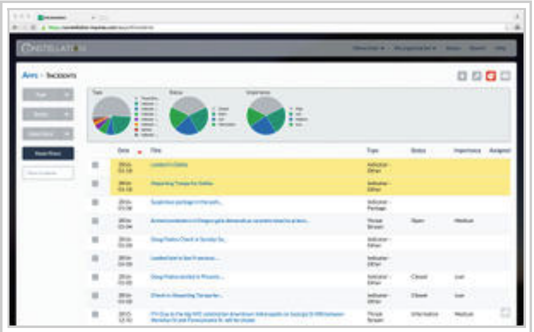

Assessment view screen with Overview section

The individual assessment view presents a list of sections on the left, and the main question content in the main area. These questions are multiple choice, single choice, text input, and number inputs. Each question may have an attached image or optional comment text. On navigation from section to section they are automatically saved, or the user may press the Save button on the upper-right to manually save. When a section is complete the user should toggle the Complete switch at the upper-right. Only when all sections are marked complete may the user press the submit button. There is a print view of an individual assessment. Assessments are read-only if they have the status of Submitted or Approved.

Administrators may approve or disapprove assessments. Disapproval moves them back into "In Progress" so the user may make changes then resubmit.

Incidents

The Incidents app allows users to log incidents.


Incidents list view

A user may create Incidents in three different ways. First, there is the '+' button on the incident list page itself. Second, the user may submit a Mobile Indicator report using the 'neutron' config code. Third, A user may create an incident from a squintem in the Threatstream app. Expand details on a squintem, click 'create incident', and the incident modal will show. If title and date are available then the appropriate fields will be filled.

The list screen allows a user to filter on incident status and importance, and search by keywords. The list is sortable by column headers, and pages to allow viewing of many incidents beyond the first ten shown. The checkbox combined with the action pulldown below the table allows a user to delete incidents.

From the list screen the user may select an incident and go to the view page. The view page has sections for Overview, Details, Photos/Documents, Map, and Annotations. On this screen the user may edit information about the incident in either the Overview or Details sections. The user may upload associated images and documents. The user may set the location of the incident. The user may add annotations about this incident and choose to make these annotations private to themselves or share with the tenant or data groups.

The roles appropriate for this app are tenant_role_incident_get, tenant_role_incident_delete, tenant_role_incident_post, and tenant_role_incident_put.

Mobile Indicator

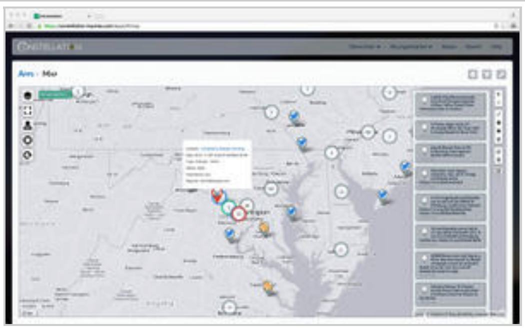
The Mobile Indicator app may be used to submit reports directly to the Constellation server. The config code of 'neutron' should be used to configure the mobile app. The Constellation username/password needs to be used. The Indicator message will be submitted to the Constellation server and permissions set such that only the tenant of the user will be able to see the message. If the user chooses to create a team on the mobile app, then a data group will be automatically created in Constellation allowing only members of that data group to see the Incidents created by the Indicator reports. These will show up on the map as Incidents but have an icon that indicates it came from the Mobile Indicator. If you join a team then the team member location will also show up on the map.

Events

TODO

Map

The map screen shows data in a geospatial view. There is the main map view area, controls on the upper-left, and controls on the upper-right.


Map view

The controls on the upper-left include: layer selector, fullscreen mode, find location, find current position, and reset view. The layer selector has three groups of layers, the base map tiles, the system data layers, and the user-added layers. By default the third group does not exist until a user adds layers. Only of of the five base tile maps may be chosen. The system data layers are elements from the other apps, such as Assets, Assessments, and Incidents. Toggling the layers on will be persisted between sessions.

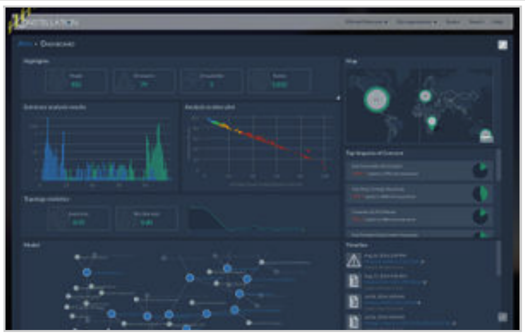
The controls on the upper-right include: zoom, annotation drawing, and the waterfall. The annotations include lines, polygons, rectangles, and points. These are shared across the tenant so other members of the team may see them. The waterfall toggle is the last controls and shows the waterfall data view. The waterfall shows a time-ordered list of items currently seen in the map view. When you move around the map or zoom in and out, the elements shown in the waterfall view will change.

Manage Layers

Clicking on the manage layers icon in the upper right above the map will take the user to the manage layers screen. This is a table of user-defined layers. The user may add a new layer by clicking on the '+' button. The supported map layer types include: DS7, MapBox, ESRI Map Service, ESRI Feature Service, Web Map Service (WMS), GeoRSS, and KML. Picking the DS7 type will require a login to a DS7 server. The ESRI types may use an authentication token.

Dashboard

The Constellation Dashboard app is meant to be a place for users to see high-level statistics, trends, data, and snapshots of their organization. The dashboard will feature a library of widgets from which the user may select, and the user will be able to interact with many of the widgets and use them as launching points into their data. Currently the dashboard has a very Insider Threat focus to it.


The dashboard

The dashboard was designed as a single-page application with one or more widgets on the screen. The widgets are in boxes that may be resized or rearranged. The size and position of these widgets may be saved per user. The user can use the settings button on the upper-right to set these preferences.

Widgets

The highlights widget shows some summary counts over the people data in the system who have been run through the Carbon model. There are four summary counts. Total population count, number of people in the bottom clearance worthy bin, number of people added to the watchlist, and total number of events processed on the entire population. Clicking on the first three numbers will take you to a filtered list on the assets page.

The histogram shows the distribution of the population by both top level hypothesis nodes. Mousing over the histogram bars will show the values (counts) for each of the two data series. Clicking on a bar will take you to a filtered list on the asset page.

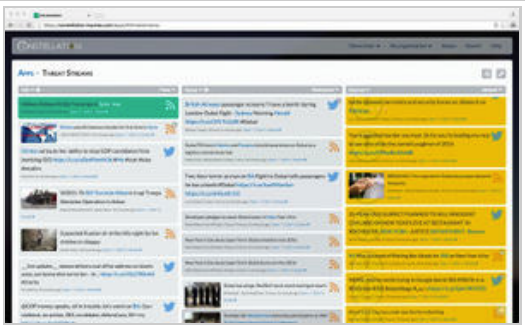
The model viewer is a read-only view of the model structure. Currently it shows a reduced model. Evidence nodes are the smaller grey ones, up to the higher-level hypothesis in the larger green node. Currently only one of the two top-level nodes are shown. This widget uses a D3 force-directed graph.

Filters

The filter button on the upper-right will bring up the filter modal popup. Here you may turn on one or more filters that will filter down the data shown in many of the widgets. However, some widgets do not respect the filters since the filters would not affect the data shown by the widget. Currently the filters available are date/time range and tags. Set the date/time range with a start and end to define the date range of data you wish to see on the dashboard. The tag filters allow you to add tags that are used to filter assets and other objects that have tags.

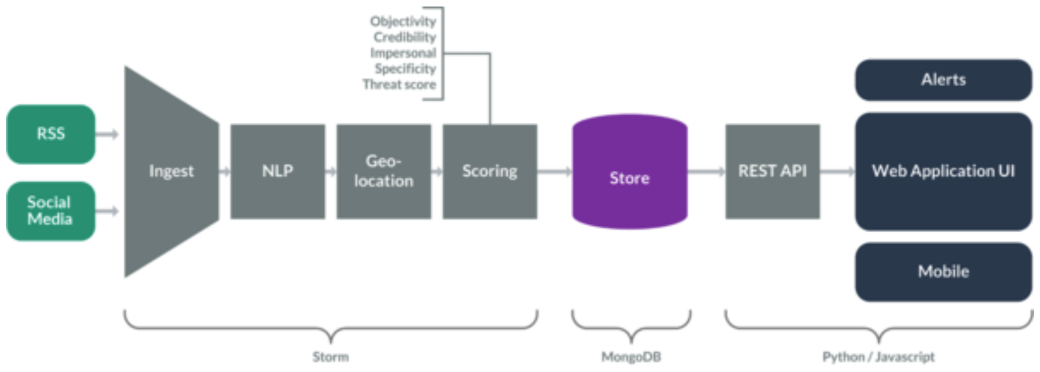
When the filter is on, a notice will appear on the upper-right next to the filter button. A small x on the filter notice will let you easily turn off all filters. Each widget respecting the filter you have set will also have a filter indicator on the widget header to show which filters are being applied to that particular widget.

Threat Streams


Threat streams three-column view

The Threat Streams app allows users to see prioritized threat-related news and social media which has been processed and scored by our analytics system.

In this app the streams are organized into three columns, and each columns provides options at the top of each that can be set up to filter data. These are situated on the right and left at the very top of the grey threat stream columns. On the right side of the Threat Stream columns a user may set filters to sort documents by calculated risk, time and urgency, i.e; Relevance, Time, Threat Model Score, Importance Score, and Alternate Score. On the left side of the Threat Stream columns a user may select to filter the Threat Streams in ways related to the type of Threat Stream. There are some preselected options to choose from; Starred, Stream, Trending Last Day, Trending Last Hour, and an 'add new' option, which will allow them to define their own filter to find threats specifically by Type, Tags, Phrases, and Location. A user can edit the filters they set by using the gear icon next to the filter type to edit them. Once a user has set up the filter the way they would prefer they can view the data that is streaming below. They can open the specific data by selecting 'Open' at the bottom of each. In doing so they will be taken to the data or threats place of origin, for example the tweet url. Users can also see details on specific streaming threats by selecting 'Details'. Within this 'Details' section users can see who the threat was from, when it was published, how long ago this data came in, it's Relevance Score, it's Alternate Score, Importance Score and it's Threat Model Score. They can save the data as Mark Important, Hide, Share, or Create Incident by clicking the blue buttons at the bottom of the 'Details' section. Creating an incident will create a new incident they can see in the Incident app.



Manage Feeds

The user may add their own feeds by selecting the grey Feed Management icon in the upper-right. The user may add RSS feeds and assign visibility so that articles ingested and processed are seen only by them (private), their groups, or their tenants. The user may also select a checkbox to suggest promoting a feed to public (system-wide). If the user adds tags to the feeds, these tags will be put on all processed articles, and may be used as a filter entry in defining a column filter on the three-column view.

Stream Processing

RSS and Twitter items are ingested constantly by the system. The RSS feeds are a collection of user and system feeds. The Twitter content is determined by a search sent to Twitter consisting of threatening topics in order to return a stream of tweets that are more threatening in nature. Approximately 300,000 documents are read in each day.

During stream processing, the text of a tweet or news item is automatically analyzed in real-time. Names of people, locations, and organizations are identified. In general, we call these these things "named entities". Upon clicking on a named entity, you will be brought to its profile page. Its profile page contains tweets/news items that explicitly mentions the named entity. In addition, the sentiment of a named entity can be viewed over time. The sentiment of a named entity is calculated during stream processing.

Each tweet/news item is also compared against a threat model. The threat model contains rules that best attempts to match a tweet/news item to a particular topic in the threat model. The tweet/news item will display tags according to how it matched against the threat model.

Mobile

Mobile Constellation is a mobile app which interfaces with the existing Constellation server API. It presents the same apps that the web version supports. A user logs in with their Constellation credentials, and is presented with Assets, Assessments, Incidents, etc depending on their roles. Screens, for the most part, mirror the functionality of the web application with data table views, single object views, and edit pages.

Available for iOS on the Apple iPhone App Store.

Menu

At the core of the app is an always-accessible menu that contains a list of apps and links to the 'What's New?' view, Settings view, and logout. In addition, the current user's email address is displayed at the top. The menu can be accessed via the menu button on the top left of the primary view or by swiping from the left edge of the screen to the right. The menu button only appears on smaller devices (e.g. iPhones and iPods) because on larger devices (e.g. iPad and iPhone 6s Plus when sideways) the menu is always open. The menu is always accessible on smaller devices by swiping from the left side of the screen, even if the menu button is not available. Additionally, the menu closes automatically when an app or link is selected, or the user taps on the main view.

Network connectivity

Offline mode works for all objects. Data is cached when appropriate where a network connection cannot be detected. When creating a new object, the app allows for internal creation and holds on to it in a local DB then attempts a server save. If save fails, the local copy remains and app is still usable. Will synchronize later to create objects up on server when network connection is back. Overall, the user does not have to think about offline vs. online. It is handled in the background. Small red exclamation marks indicate to the user where an object hasn't been synchronized up to the server.

Login

When the app is first launched, the user is prompted to login with his or her email and password. After entering the required information and tapping the login button, a spinner will appear in place of the login title to indicate network communication is taking place. When the login is complete, the login modal will dismiss and the user will be presented with the 'What's New' view. However, if the user did not provide the required information or if the information was incorrect, an alert will popup and the login modal will not dismiss.

If the user is logging in for the first time with an account that has not been used on the mobile device before, the app will initialize with all the data from the server. This can take a moment if a large amount of data is being delivered. The initial data setup is a one time action per user account. If the user resets the app via settings, the initial setup will occur again, because all the data was erased.

Anytime the user's login session expires and the user attempts to use the server, the login modal will appear and the user must login to continue use. However once logged in, the user's session is restored to where they last were.

Authentication / Authorization

The Constellation platform has multiple client domains, or tenants, in a single platform instance. This means all data and users are co-existing in one instance, but permissions and security keeps the data and users separate in each tenant. Users have tenants, roles, and data groups. Tenants are like the client organization in which all users exist. Within the tenant there may be data groups to segment what can be seen by different groups. Roles dictate what apps and permissions users have, such as read-only access to assets, or full edit access to Incidents.

A user may have a tenant administrator role, then per app there are four roles (create, read, update, delete). Tenant administrator has control over managing users in their tenant.

These roles and data groups match to security tags on all objects. Objects in Constellation have a security tag with readers and writers arrays.

```
	"security" : {
		"readers" : [
			"demo"
		],
		"tenant" : "demo",
		"writers" : [
			"demo"
		],
		"createddate" : ISODate("2014-06-16T18:31:06Z"),
		"modifieddate" : ISODate("2014-06-16T18:31:06Z")
	}
```

Tenant groups and users can hold settings such as theme and other configurations. These exist off of the tenant for client-wide customization, or can exist on user accounts themselves for user-specific settings.

On the cloud-based platform, we use a third-party resources for authentication and authorization. For on-premisis installs, there is the ability to use a local database to store accounts and role information.

User Management

Users can sign up themselves by registering at constellation.haystax.com. When they register themselves, a new tenant is created just for them and they are made the administrator of that tenant. Users may then invite new users into their tenant. When an invitation is accepted that new user will automatically be in that existing tenant instead of creating their own. If a tenant desires self-registration to require administrator approval, that may also be turned on to make sure the admin reviews all new accounts before they may log in.

User accounts may be created ahead of time or users may self-register. Users use their email addresses as usernames. Users have roles, data groups, and a tenant. All users can have multiple roles, be in multiple data groups, but only be in one tenant. Roles determine which apps the user can access, and also dictate the form of access (read, write, create, delete). There is a specific role for administrator access. Data groups define the partitioning within a tenant which controls data each user group can see (think districts within an entire state tenant). The administrator can manage access of the other members of their tenant using the "Manage access" option in the upper-right.

API

How to use the API as a third-party.

```
https://constellation.haystax.com/serverdocs/
```

Deployment and System Architecture

How to deploy on one machine, multiple machines, scaling concerns, database replication, failover, etc.

Constellation is installed on Linux machines. RedHat/CentOS 7.2, Ubuntu 16.04, and macOS Sierra are supported.

Constellation uses MongoDB no-SQL data store for storing all data and results.

TODO

Installation

Fusion is one part of the entire Constellation stack. It's the part that does the processing of data to map to a Bayes model and run for belief results. The UI stack is separate, and may be installed on separate machines. The UI stack is sometimes referred to as "Neutron" as its code name. The Neutron stack has two API layers and a UI layer. The Fusion stack is all Python. Both use MongoDB as their data stores.

There are two installers. One for Fusion. One for Neutron. The database install below uses the Fusion installer. The installers work on RedHat 7.2, Ubuntu 16.04, and MacOS Sierra.

To install the application at a remote (customer) site from packages it is recommend that you have at least 3 machines. Everything can all be installed on one machine.

- Database
- Fusion (Computation)
- Neutron (Web)

You need `fusion.YY.tar.gz` and `neutron.ZZ.tar.gz`. The YY and ZZ are the fusion and neutron release versions respectively.

Database

To install the database follow these steps:

1. Copy the fusion.tar.gz package to the machine.

2. Create a directory to untar the package into and then untar in that dir (assuming the package is in the user's home directory)

```
$ mkdir db
$ cd db
$ tar -xzf ~/fusion.tar.gz
```

3. Install the package using make

```
$ cd fusion/constellation.configuration
$ sudo make install-mongo
```

Neutron (UI)

Neutron is the code name for the API, Server, and UI layers that make up the Constellation interface. Neutron can be installed without Fusion. To install Neutron follow these steps:

1. Copy the neutron.tar.gz package to the machine.

2. Untar the package.

```
$ tar -xvzf neutron.tar.gz
```

3. Install the package using make

```
$ cd neutron
$ sudo make install
```

When prompted type in the ip address of the database machine. If everything is on one machine use 127.0.0.1.

4. Verify the interface works

```
http://localhost/
```

Fusion

Fusion is the back-end processing system which applies data and evaluates a model to produce scores on assets (people). Fusion can be installed without Neutron. To install Fusion follow these steps:

1. Copy the fusion.tar.gz package to the machine.

2. Untar the package.

```
$ tar -xvzf fusion.tar.gz
```

3. Switch to the constellation.configuration directory.

```
$ cd fusion
$ cd constellation.configuration
```

4. Install the package using make. When prompted type in the ip address of the database machine. If everything is on one machine, use 127.0.0.1.

```
$ sudo make install
```

5. Run the tests. The tests are run using pytest package. If any of the tests fail, you will see error messages in red. Keep in mind that test_run.py takes ~8 min to execute, so be patient.

```
$ make test
```

Running the fusion system

1. Modify config.json to point to fusion_share folder for data and data_tranform files

```
$ cd ~/fusion/constellation.configuration/constellation
$ sudo vi config.json
```

Under local_carbon, set

```
"load.source.path": "/location/of/data/data"
```

```
"transform.source.path": "/location/of/data/data_transform"
```

Note: sample CSV data files and transform files can be downloaded. Set the above directories to the location of the data and data_transform directories for your particular data files.

2. Execute run.py. Remember to modify run.py args as needed. Example run.py command below runs data through Fusion.

```
$ cd ~/fusion/constellation.configuration
$ python constellation/run.py -e local_carbon -t demo -pf /location/of/data/data/HRData.csv -pc 'Employee Identifier' -i csv
```

Please be patient. *Depending on the size of the data, it might take some time to see output from the script. In future releases, we will add status messages.*

Log files

Log files can be found in `/var/log/constellation`.