Technical Report: DAVE February 2020

Data Analytics and Visualization Environment for xAPI and the Total Learning Architecture

25 February 2020

This work was supported by the U.S. Advanced Distributed Learning (ADL) Initiative HQ0034-19-F-0385. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ADL Initiative or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
02-25-2020	Research Report	February 2018 - February 2020

4. TITLE AND SUBTITLE

Data Analytics and Visualization Environment for xAPI and the Total Learning Architecture: DAVE February 2020 Technical Report

5a. CONTRACT NUMBER

HQ0034-18-C-0040

5b. GRANT NUMBER

N/A

5c. PROGRAM ELEMENT NUMBER

0603769D8Z

6. AUTHOR(S)

Blake-Plock, Shelly.

5d. PROJECT NUMBER

N/A

5e. TASK NUMBER

N/A

5f. WORK UNIT NUMBER

N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Yet Analytics, Inc.
201 E Baltimore Street
Suite. 630
Baltimore, MD 21202

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

OUSD Personnel & Readiness
Advanced Distributed Learning Initiative
13501 Ingenuity Drive, Suite 248
Orlando, Florida 32826

10. SPONSOR/MONITOR'S ACRONYM(S)

OUSD/P&R/FE&T/ADLI

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution A

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report summarizes the culmination of the Data Analytics and Visualization Environment (DAVE) project. This report focuses on all aspects of the projects that are included in the beta release of February 2020. This report lays out a technical summary, software updates, deliverables, and a wayahead. Software components, access information, user interface, and user experience are all covered in this report. In addition, data specifications are provided, along with technical information about the portability of data. The beta state is described, and the pathway to a final prototype is given, as well as supplemental efforts that would benefit DAVE.

15. SUBJECT TERMS

Data Analytics, Data Algorithms, Primitives, Data Visualizations, Analytics, Algorithms, Visualizations, ADL, Advanced Distributed Learning, DAVE, Data Analytics and Visualization Environment

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Nick Armendariz
U	U	U	UU	17	19b. TELEPHONE NUMBER *(Include area code)* 407-381-5550

Reset

Technical Report: DAVE
February 2020

Data Analytics and Visualization Environment for xAPI and the Total Learning Architecture

Prepared by Shelly Blake-Plock, PI, Yet Analytics, Inc.
Submitted to ADL TPOC on February 25, 2020

1. Purpose of this Document

The purpose of this document is to present a report and related summary documentation regarding the beta prototype of the Data and Visualization Environment (DAVE) for xAPI and the Total Learning Architecture.

2. Project Description

The DAVE Framework aims to provide a common means through which DoD and Federal Stakeholders can analyze, interpret, and visualize micro-level behavior-driven learning aligned to the technical requirements of xAPI, xAPI Profiles, and the Total Learning Architecture (TLA).

DAVE provides a user with the ability to customize xAPI data analysis and the resulting portable data visualizations.

The framework itself was built to be deployed anywhere (on the JVM or Javascript runtimes) in Clojure and ClojureScript. It is intended to require very little setup and configuration by a user.

DAVE is available as an Open Source Apache 2.0 licensed project on Github <https://github.com/yetanalytics/dave>.

This beta implementation of the DAVE Framework builds on the alpha implementation and adds what was discovered in earlier phases of the project in order to provide the required flexibility for broad applicability across DoD and Federal Stakeholder needs.

The fundamental advancement of this update to the DAVE Framework is the addition of the ability for a user to develop a custom Analysis of xAPI data without having to extend the framework itself.

While earlier iterations relied on research Questions, this approach was found to be inadequate

for government needs. In the research question model, while some flexibility existed for the development of resulting visual elements, the filter and transformation of the data itself (referred to as a Function) had to be pre-defined as part of the application — in essence, limiting end-users to what had already been considered unless new code was added to the Function library.

Upon completion of this beta update to the Framework, DAVE now has these new key features:

- **Functional Primitives** which break the processing of xAPI data down into discrete units that can be composed to create new analytical functions. Primitives allow users to address the methodology of answering research questions as a sequence of generic algorithmic steps which establish the necessary data transformations, aggregations, and calculations required to reach the solution in an implementation-agnostic way.

- **Statement Filtering** which allows the filtering of xAPI statement collections against any criteria relevant to research goals. This allows a user to ignore data that is irrelevant to their problem domain.

- **Transform Language** which combines data filtering and data transforms (as defined in the Primitives definition) into a coherent syntax that can be used to efficiently instruct the Dave Framework on how to treat xAPI data.

- **UI Improvements** which have simplified the navigation and user interaction with the system.

- **Documentation** which includes the update of the algorithm description for the use of functional primitives.

3. Technical Summary

DAVE — the Data Analytics and Visualization Environment for xAPI and the Total Learning Architecture — provides a means of quickly creating portable data visualizations purpose built for xAPI-based semantics. The DAVE Framework provides a language and syntax for filtering and transforming xAPI data and a path to output it into completely customizable visualizations.

The architecture of DAVE, and the technologies and concepts employed for implementation, satisfy two primary deployment considerations. First, this beta version of the application is lightweight and can be deployed in any location, including browser-only with no server side implementation. This makes it portable and requires very little — if any — setup for basic use.

The other consideration is potential scale. The technologies and protocols chosen for these beta additions to the DAVE Framework can be scaled to handle very large streaming datasets, and can be modified to distribute workloads horizontally.

All of the source code and documentation supporting the DAVE framework is located as an Apache 2.0 licensed project on Github <https://github.com/yetanalytics/dave>.

The framework provides scaffolding for the design of algorithms meant to leverage the attributes found in xAPI data statements. These templates — as well as a basic rendering of xAPI in Z Notation — are located in a master doc <https://github.com/yetanalytics/dave/blob/master/docs/main.pdf>.

Yet Analytics implemented the concepts described in the algorithm documentation in the Clojure(Script) and Datascript languages.

DAVE runs on both the Java Virtual Machine (JVM) and JavaScript environments.

The DAVE data model and object hierarchy was implemented in Clojure(Script) and the unit and generative testing facilities were implemented in Clojure(Script).

Yet Analytics generated multiple datasets using the Data and Training Analytics Simulated Input Modeler (DATASIM) to aid in the iterative creation and testing of the product and in the release of the reference model. This simulation tool was designed to generate realistic xAPI Statement datasets from a conformant xAPI Profile, a set of Actors, and alignments between the Actors and components of the Profile.

The DAVE user interface includes:

- a **library** (which is the user's homepage and which contains all of the user's workbooks)
- **workbooks** (which contain all of a user's analyses)
- and **analyses** (which include query code, visualization code, and portable data

visualizations)

The beta version of the interactive front-end includes viewable components for the DAVE object hierarchy and schema. The user is provided access to the data source. And as part of the new Analysis editor, both the data transforms and visualization specifications are available in the reference UI.

As of the beta release, DAVE workbooks include user interface elements to allow CRUD (create, read, update, and delete) operations on DAVE workbooks and on DAVE objects.

Yet Analytics implemented an HTML5 browser path navigation against the DAVE hierarchy, implemented a user “toast”-style notification system, and implemented live instrumentation testing of DAVE functions in a browser environment.

As a feature of the Analysis engine, validation and related feedback from both transform syntax and visualization specification inputs are available in the reference UI. Live debugging facilities were added to the DAVE browser development environment and were implemented as browser local storage persistence for DAVE workbooks with schema validation.

The updated beta user experience of the dataflow graph follows the rule of: Data Source --> Analysis, whereas the user chooses a data source, and is brought to an interface to edit both the Transforms and Visualization. In previous iterations of the Framework, the flow was Data Source --> Function --> Visualization, and the Function and Visualization were contained inside a Question. The concepts of Function and Question have been deprecated.

Yet Analytics implemented a dynamic chart display using the open source Vega visual grammar specification and the vega.js rendering API.

A base version of the UX/UI styling and visual language from the DAVE user experience mockups was applied to the prototype and a JavaScript build suitable for use on Github Pages was deployed during alpha in late December 2018 and is the current environment of the DAVE reference model.

4. Accessing DAVE

The DAVE beta prototype will be available at https://yetanalytics.github.io/dave/ in February 2020. Users will be able to access DAVE without the need for an account, as it runs entirely in the user's web browser. Changes users make are saved to browser local storage, so they are persistent.

Developers wishing to run the DAVE development environment (or the tests) should see the project README: https://github.com/yetanalytics/dave/blob/master/README.md.

5. Reference Datasets

In order to assess the usability of DAVE, sufficient amounts of relevant xAPI Profile-conformant xAPI data are required. For this reason, the Data and Training Analytics Simulated Input Modeler (DATASIM) was used to generate xAPI data from multiple profiles. This data was ingested and processed by the DAVE Framework to produce visualizations and draw conclusions.

As a baseline, and for high volume data generation, cmi5 was chosen. This allowed us to use a commonly applied Profile to show DAVE assessment use cases and test DAVE's current performance capabilities.

For xAPI Profile Analysis through DAVE, we used data generated from the Tactical Combat Casualty Care Course (TCCC) Profile Subset.

The applicable subset of the TC3 xAPI Profile can be found here:

https://github.com/yetanalytics/datasim/blob/tc3_profile/dev-resources/profiles/tccc/cuf_hc_video_and_asm_student_survey_profile.jsonld

The TCCC CUF HC xAPI Profile is

- scoped to TCCC CUF HC and TCCC ASM content available from the deployed medicine website

- partially derived from the xAPI Video CoP profile
- designed for programmatic generation of diverse xAPI datasets; datasets which include a multitude of emergent and repeatable patterns of behavior; behavior modeled within this profile as statement templates and patterns.

6. Beta Implementation

6.1 Analysis

In previous iterations of the DAVE Framework, data transforms and the filtering of xAPI data was conducted through the use of a “Question”. The Question was tied to a specific predefined algorithm — a Function — which could be reused. An option for expansion in that model is that this list of Functions could be expanded to include many practical use cases, but the model requires code to be added to DAVE to perform the transforms. In this way, at its core, the framework was not dissimilar from a traditional analytics dashboard solution.

In this beta update, the desire was to break away from that model entirely and empower the end user to enact their own functions on the fly. By putting the power of filtering, transformation, and aggregation at the user’s disposal, this interaction allows a user to account for any possible use-case.

As a result, the concepts of “Question” and “Function” were deprecated in DAVE in order to remove ambiguity. The new concept being introduced has been named “Analysis”. An “Analysis” consists of a Transform input and a Visualization input, each in their respective syntaxes.

The old interaction, from a user perspective, followed a “Wizard” design pattern, wherein the user picks various options along successive pages until a result is generated. The new model follows a pattern much more analogous to a REPL (read-eval-print loop), as the output is shown beside the evaluated inputs and iterative small changes can be seen by modifying the inputs. The resulting user experience is much more conducive to the prototyping of data visualizations.

6.2 Visualization Syntax (Vega)

While the work performed for the DAVE Framework does not prescribe a specific visualization engine for all applications of DAVE, it was necessary for implementation of the Beta and for a

demonstration of framework capabilities to select a compatible visualization engine. For the Visualization Syntax input to the Analysis concept, the requirements are as follows:

- Accept data inputs compatible with DAVE transform outputs (JSON)
- Generate visualizations on-the-fly, in-browser, with a portable specification
- Generate a large variety of visualization types, and offer specification-based options for the comprehensive customization of each visualization type

For these reasons, for the beta update to the DAVE Framework, Vega[1] was selected as the Visualization Syntax. The Vega project is described as “a visualization grammar — a declarative language for creating, saving, and sharing interactive visualization designs”. Vega’s inputs are a JSON-based visualization syntax, and JSON-based data inputs. The outputs are visualizations in Canvas or SVG. Refer to **Section 9.2** for a generic example of the Vega visualization syntax.

6.3 Data Transformation Concepts (DAVE Primitives)

The next part of updating the DAVE Framework from a collection of Functions into a utility capable of producing data outputs and visualizations for any learning analytics needs is the addition of some conceptual components establishing a common language for xAPI data transformation.

The shape that these components take is Operations, Primitives and Algorithms. Primitives are comprised of Operations and Algorithms are comprised of Primitives. The focus of this work was to define the properties of and interactions between Operations, Primitives and Algorithms in a general way which doesn’t place unnecessary bounds on their range of possible functionality with respect to processing xAPI data.

Many of these transforms (especially Operations) are already implemented in programming and/or query languages because they are so essential, though it may be by a different name.

For these conceptual parts of the DAVE Framework, Z Notation was used to separate the transform itself from any given language or proprietary representation.

[1] https://vega.github.io/vega/

With these building blocks, an analyst can take an xAPI dataset and answer a new research question without having to rely on previously-defined Functions. They may then visualize the output in the DAVE Framework.

6.3.1 Operations

An Operation is a function of arbitrary arguments and is defined using Z Notation. The use cases of these operations are explored in **Section 4** of the accompanying *DAVE Learning Analytics Algorithms* specification.

An example of such an Operation is *first* which, as its name implies, takes the first element from an ordered collection of elements. If we ran *first* on the following collection:

[“A”, “B”, “C”] -> *first* -> “A”.

Some other common Operations include:

first	rev	distributed concatenation
second	head	disjoint
succ	last	partition
min	tail	bag scaling
max	front	bag union
count	extraction	bag diff
concatenation	filtering	items

6.3.2 Primitives

Primitives allow users to break down the transformations, aggregations, and calculations required to reach a solution into discrete steps. Primitives are composed of Operations and other Primitives. They can be chained together to produce a desired Algorithm.

An example of a Primitive is *Get[V, Collection]*, which is a retrieval of a *V* located at *id?* within *in?* where *in?* can be a Collection or Key-Value.

6.3.3 Algorithms

An Algorithm is a composition of Primitives that accomplish a specified analysis goal. Algorithms have component steps, each being defined by Primitives and Operations, and the steps are meant to be a guide to how to look at data transforms in an xAPI context. For an example of a fully explored Algorithm refer to **Section 9** of the accompanying *DAVE Learning Analytics Algorithms* specification.

6.4 Data Transform Input Syntax (Datalog)

While Operations, Primitives and Algorithms define an implementation-agnostic way of describing transformations and aggregates of xAPI Statements, they do not fully address the practical matter of implementation syntax.

The Z Notation format of Primitives works well for defining capability, but inputting Z Notation into every query is cumbersome and unintuitive to an end user. Furthermore, existing filter and transform languages could be expanded to include xAPI Primitive functionality while retaining their existing capabilities. As mentioned, most programming and query languages implement the majority of the base Operations out of the box.

The unique challenge presented by this part of the DAVE Framework is that while many existing database protocols fit the needs of the project from a syntax perspective, they are typically tied to their proprietary database storage format. It would, for example, be straightforward to persist xAPI statements in a specific database engine and then leverage the native query protocol of that engine to perform similar transforms and operations directly on that database.

Because of the JSON format of xAPI, the research on this part of the implementation of transforms initially led to investigation of existing JSON-based document database formats and protocols (including MongoDB and CouchDB). The limitation of that approach came down to cross-LRS compatibility and the existing xAPI specification. The data would be required to be in a proprietary storage format in order to use existing parsers and engines, and communication with that data would have to be over proprietary protocols.

The correct solution instead calls for working with a sufficiently robust syntax, with room for Primitive extension, and harnessing its power to perform transforms on any xAPI data stream from any LRS. The reason this last part is so critical is that the xAPI specification defines a standard for the retrieval of Statements over REST. If DAVE is to be flexible enough to have broad applicability across the DoD — or across any complex enterprise — it must be able to optimally handle a stream of xAPI statements from any source.

This research led to the implementation of Primitives within Datalog[2].

A Datalog query is declarative, logic-based, and runs rules and operations on a database of facts. Datalog databases also commonly have the ability to express complex nested relationships like those found in xAPI statements. Traversal to a specific field deep within a statement, for instance, can be expressed as a rule and be reused in multiple queries.

The DataScript[3] implementation of Datalog features the creation of easy to use custom functions, both for filter and aggregate parts of the query. This feature makes implementing Primitives as reusable functions within the Datalog syntax straightforward. What was implemented for the DAVE beta implementation was an adaptation of DataScript made specifically to process streaming xAPI statements. A big advantage is that it was adapted with primitives built right into the language as callable functions.

7. Next Steps: Getting to TRL 6

This beta update of the DAVE Framework includes a live working and tested web application for reference modeling purposes. It fully harnesses the power of completely custom transforms joined to completely custom visualizations and provides a path for almost any

[2] https://clojure.github.io/clojure-contrib/doc/datalog.html
[3] https://github.com/tonsky/datascript

analysis possible on conformant xAPI Statements in any LRS. In the ability to generate the full range of possible analysis on xAPI data, DAVE achieves a level of technical usability commensurate with TRL 5.

In moving to TRL 6 and transition into the operational environments beyond, it will be necessary to increase that usability for a more generalized business audience.

The alpha iteration of DAVE employed a guided wizard interaction which, while opinionated and limiting in capability, did allow for the execution of stock Functions without the user having intimate knowledge of the technologies at work. This beta iteration of the technology made a tradeoff for vastly increased analytical capabilities at the expense of push-button usage.

As it stands currently, the platform is geared toward analysts with at least some understanding of the underlying data. As the platform is used and user interactions and pain points are observed, the platform should evolve to aid user experience in its design. There are a number of ways this can be accomplished.

- Editor tooltips and advancements in the transform and visualization editor screens may be evolved to add code completion and contextual suggestions
- Syntax guides may be made more readily available at a user's disposal
- A shared stakeholder library of query and visualization templates categorized by domain and problem set could be created
    - This solution would fully leverage the Linked Data capabilities offered by xAPI Profiles and supported by the xAPI Ontology and Profile Validation resources — increasing the transition potential of every piece of the TLA that either contributed to or took advantage of the resource set
    - This solution could vastly speed up the time-to-insight for learning analytics and reporting across the DoD enterprise

The latter is the most useful way to increase platform usefulness, decrease the learning curve, and to make the DAVE Framework more broadly applicable to DoD and Federal Stakeholders. Two applications of this concept are explored in more detail in **Sections 8.1 and 8.3**.

8. Opportunity for Further R&D

The last two years of DAVE research into xAPI data analytics has also uncovered some needs unique to analytics capability and innovation within the context of the Total Learning Architecture. The key (current) missing pieces in data analytics for the xAPI ecosystem for DoD stakeholders include:

- the analytics template library mentioned above and the requisite validation tools for the xAPI Profiles upon which the templates are based
- a reference architecture for the high volume scaling of “Big xAPI Data” analytics
- rule bundling for reusable xAPI analytical operations

With the inclusion of this work, DAVE becomes a complete suite of analytics and visualization specifications and reference models which extend xAPI and Total Learning Architecture capabilities to DoD ADL stakeholders and the enterprise data analytics space.

8.1 DAVE Analysis Template Library & xAPI Profiles

This update to the DAVE Framework has provided users with the ability to create a completely new and custom Analysis for xAPI data from any LRS. This capability for custom filter and aggregation over predefined functions opens the door for the sharing of Analysis templates which can be reused and distributed across DoD and Federal Stakeholders.

The creation of DAVE Analysis templates and the associated infrastructure and Library for DAVE users to build, share, edit, and repurpose DAVE Analyses will have the effect of rapidly increasing the usefulness and flexibility of the technology. These analyses are performed against data shaped by the Templates and Patterns of one or more xAPI Profiles.

Association with xAPI Profiles creates a natural path to reuse for a particular Analysis and because xAPI Profiles are used to validate incoming statements, they can validate relevance to a particular form of Analysis. Rather than always starting from scratch, or from a few centrally developed stock templates, an individual analyst would have the ability to search the Library for templates specific to the kind of reporting they need for a given project, primarily based on which xAPI Profile Patterns and Templates their data is aligned to.

Additionally, in order to fully leverage xAPI Profiles as a guide to Analysis within the DAVE Template Library, it is required that there be a consistent methodology to validate Statement Templates and Patterns within xAPI Profiles. Currently there is no tool capable of traversing an xAPI Profile to validate its conformance to the xAPI Profile specification. Furthermore, it has been discovered that aspects of the current xAPI Ontology are broken and would make a validator unable to leverage JSON-LD's capabilities for traversal and validation of xAPI Profiles. It is recommended that these issues be resolved in order to enable a properly Profile-aligned Analysis Template Library.

8.2 High Volume Transform Scaling Reference Architecture

The architecture of this update to the DAVE Framework allows for the processing of large streams of xAPI data and the resulting visualization of the output. The transforms are executed through an adapted query processor which uses intelligent memory management to execute Primitive operations on non-Datalog data source streams (which is why it is compatible with any LRS). This capability could be expanded to allow scaling with any input data size. A combination of intelligent memory management, the application of intelligent filters, and a horizontally scaled parallel processing model has the potential to process the most extreme theoretical volumes efficiently without going around the existing xAPI communication specification.

8.3 Rule Bundles

The syntax for the expression of DAVE Primitive transforms and Algorithms is a Datalog variant and one of the major benefits of such a syntax is the use of Rules.

Within Datalog, and by extension within the DAVE transform syntax, Rules represent a collection of operations that are bundled to be reused within an expression. The distinction here is that rather than just templating and sharing the query and visualizations at the top level, various internal techniques for filtering and aggregation specific the the xAPI community could be made into reusable components. This would facilitate both higher productivity for analysts, and a path to learn the analysis techniques themselves to apply to other analyses.

9. Appendix

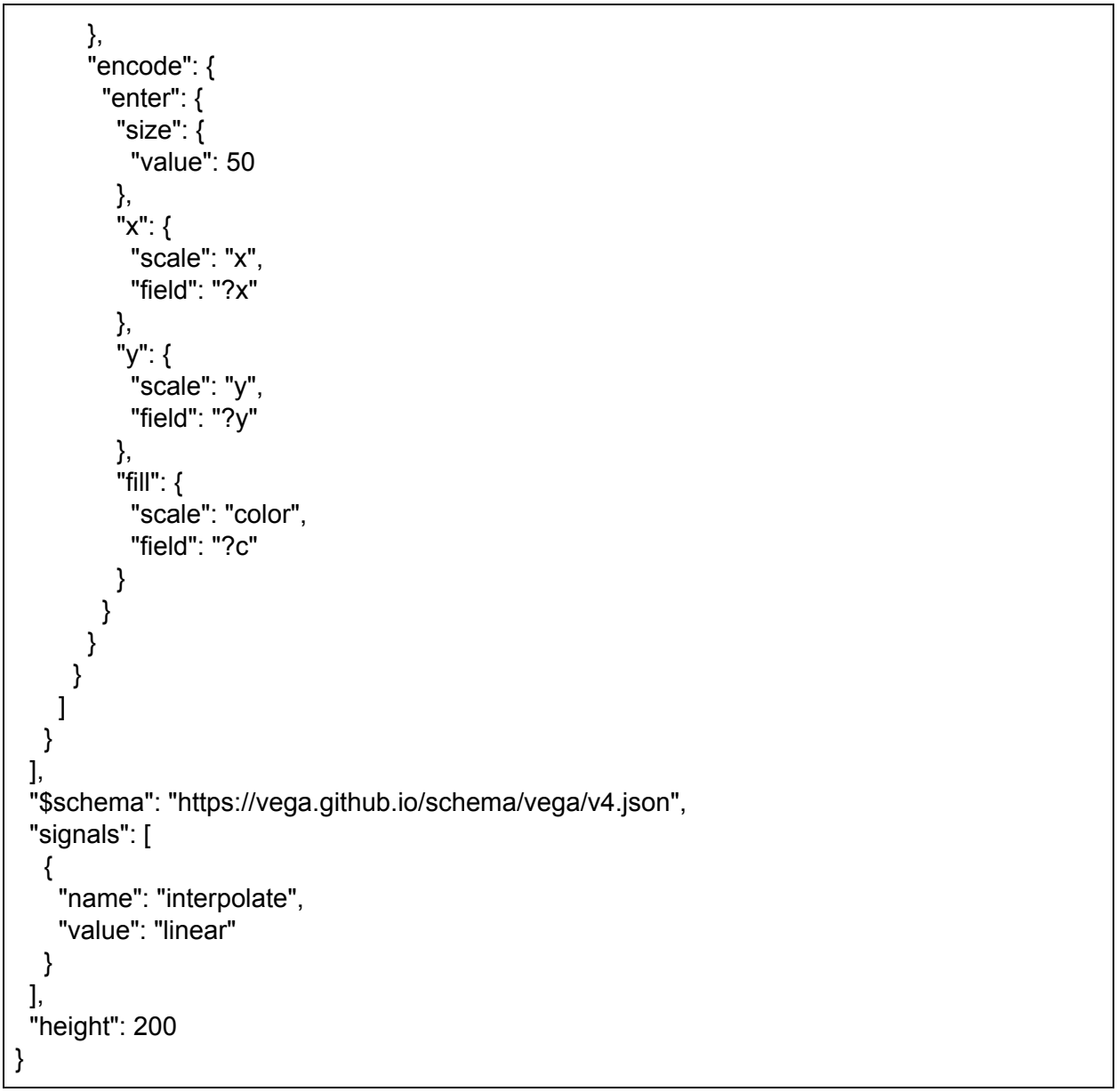
9.1 DAVE Live Reference Implementation

To access the a DAVE Beta live reference implementation, proceed here https://yetanalytics.github.io/dave/. This deployment runs directly in a browser, and relies on local browser storage, so compatibility issues should be minimal for any users wishing to explore the implementation.

9.2 VEGA Syntax Example

```
{
  "autosize": "fit",
  "legends": [
    {
      "fill": "color"
    }
  ],
  "axes": [
    {
      "orient": "bottom",
      "scale": "x",
      "labelAngle": 60,
      "labelAlign": "left",
      "labelLimit": 112,
      "labelOverlap": true,
      "labelSeparation": -35
    },
    {
      "orient": "left",
      "scale": "y"
    }
  ],
  "width": 500,
  "scales": [
    {
      "name": "x",
```

```
    "type": "time",
    "range": "width",
    "domain": {
      "data": "result",
      "field": "?x"
    }
  },
  {
    "name": "y",
    "type": "linear",
    "range": "height",
    "nice": true,
    "zero": true,
    "domain": {
      "data": "result",
      "field": "?y"
    }
  },
  {
    "name": "color",
    "type": "ordinal",
    "range": "category",
    "domain": {
      "data": "result",
      "field": "?c"
    }
  }
],
"padding": 5,
"marks": [
  {
    "type": "group",
    "from": {
      "facet": {
        "name": "series",
        "data": "result",
        "groupby": "?c"
      }
    },
    "marks": [
      {
        "type": "symbol",
        "from": {
          "data": "series"
```



9.3 DAVE Query Syntax Documentation

The basics of the DAVE Query language, and all of the provided xAPI functionality, are documented on github here.

From that document, you can learn how Datalog works, learn how to address xAPI attributes in a query, and how to apply math functions to a transform.