VIRGINIA DEPARTMENT OF HEALTH

CENTERS FOR DISEASE CONTROL AND PREVENTION

A PROTOTYPE OF MODERNIZED PUBLIC HEALTH INFRASTRUCTURE FOR ALL:

FINDINGS FROM A VIRGINIA PILOT

—

NOVEMBER 2022

EXECUTIVE SUMMARY

A cross-functional team of public health practitioners and technologists from the Centers for Disease Control and Prevention (CDC) and United States Digital Service (USDS) began working with the Virginia Department of Health (VDH) in the fall of 2021. The team engaged in discovery in October and November, working to establish an understanding of the current state workflow at VDH, from the receipt of public health data through processing to analysis. In January of 2022, the CDC and USDS team and VDH began a formal partnership through which three data streams—lab reporting, case reporting, and vaccines—were shared to a cloud database hosted by the CDC. Between January 4 and September 30, the development team worked on updating and streamlining data ingestion and related processes. This effort resulted in a prototype of a customizable, cloud-based data pipeline comprised of a set of modular, open-source tools that process raw datasets (lab results, case reports, and vaccination records) in a single pipeline. Within this system, data is standardized, geocoded, deduplicated, and linked to better facilitate patient and case-level analyses and action. The prototype saves time and manual effort, increases data processing speed, creates a single source of truth for incoming data, and removes the need for duplicative processes. Using the lessons learned from prototype development and implementation, as well as myriad other findings from this pilot project, CDC is further investigating how to apply the principles and practices of industry-standard data architectures and human-centered design to the modernization of public health data and information systems.

AUTHORS

Molly Murray, Bryan Britten, Jeremy Zitomer, Ann Margaret Millspaugh,
Dan Paseltiner, Brandon Mader, Spencer Kathol, Celeste Espinoza,
Ryan Gaddis, Matt Goldberg, Michael Judd, & Nat Hillard

ACKNOWLEDGEMENTS

A heartfelt thank you to the Virginia Department of Health: Tim Powell, Rebecca Early, Merylyn Huitz, Sara Gowda, Greg Dennis, Prabhat Kumar, Dheeraj Katangur, Anup Srikumar, and many others. The whole team—and particularly Tim and Rebecca—spent precious time and resources on continued collaboration and selfless participation in this work. They did this work, and shared their data, not with the end goal of bettering their own system and process in the short term, but instead with the long-term vision of improving public health data infrastructure for every jurisdiction nationally. They gave the team of engineers the data and insight needed to re-design the public health data pipeline as the first step on a multiple-year journey of re-envisioning public health data architecture, analysis, and action. We are so thankful for their altruism, and for being the proving ground for this necessary work.

TABLE OF CONTENTS

INTRODUCTION AND BACKGROUND

Data coming into state, tribal, local, and territorial (STLT) public health agencies is messy, in multiple formats, lacks standardization, and often incomplete. Processing inconsistent and unstandardized data so that it can be used for action (e.g., contact tracing, case investigation) or analysis (e.g., mapping hot spots, identifying gaps in vaccinations) is a manual, time intensive labor that often results in duplicative processes, systems that time out due to data volume, and no single source of truth. Virginia had to build complex business rules and design bespoke processes to move data through the pipeline and make it actionable. This is both time and resource intensive, involving multiple staff persons, numerous systems, and even an Excel spreadsheet that scheduled out the timing of when different incoming data streams could be processed to avoid overloading.

WHAT WE LEARNED

Bringing disparate data streams together into a single database, standardizing data elements, converting to the Fast Healthcare Interoperability Resources (FHIR) standard, and then processing the data at once, further upstream in the data pipeline, resulted in:

- **A single source of truth for incoming data**
- **Faster processing times**
- **Event-triggered processing**
- **Scalable [cloud] infrastructure that can handle increases in volume of data**
- **Ability to deduplicate and link records both within and across data types**
- **Insight into performance**

Additionally, this singular pathway better allows for customized analytics after processes have been completed, resulting in more accurate reporting and representations. Furthermore, by constructing this data pipeline out of discrete “Building Blocks” (modular software services that each accomplish one specific task, that can also be combined to create larger data processing and analysis pipelines), we create an architecture that is:

- **Flexible and modular**
- **Easy to update**
- **Easier to understand**
- **Easier to make open-source**
- **Easier to integrate with STLTs’ existing systems**
- **Performs similar operations consistently**
- **Easier to share and reuse across STLTs**

WHY IT MATTERS

Today, many public health departments design bespoke solutions and use siloed systems to process incoming data streams. Often it is a manual, time consuming process to combine and link across data streams and takes additional time to move incoming data into surveillance and reporting systems for action—conducting contact tracing, case investigation, identifying break-through COVID cases, and more. The prototype data pipeline described above saves human time and effort, increases the speed with which data is processed, creates a source of truth, and removes the need for duplicative processes. Additionally, the standardized data, in a single format, allows for the creation of data marts[1] on the fly, allowing epidemiologists and other public health officials to focus in on the data they need to do their jobs and keep their communities healthy.

> **"Our epidemiologists waste 80% of their time cleaning data and can't do useful analysis. The end goal of all this infrastructure is to free up that 80% of their time to do actual public health work."**
>
> *— Public Health Official*

WHAT'S NEXT?

CDC and USDS worked with Virginia for another quarter, using real-time data to test the Building Blocks developed in the previous six months, as well as dove further into designing flexible, user-specified data marts and collecting additional lessons on the benefits and drawbacks of using a FHIR server[2]. The team also worked with VDH on implementing the prototype data ingestion pipeline environment, with live data. This prototype can serve as a parallel processing pipeline for Virginia to use as they see fit. Beyond VDH, the data pipeline and analysis tools developed above can be iterated and improved upon by being tested on additional data, in more states, to ensure that they meet the needs and improve processes on a national level.

Additionally, while these new tools were tested with a time-limited window of inbound data, in the long term, CDC could work with STLTs to test these tools with historical data and in concert with systems already in production, as well as to prioritize and develop additional modular Building Blocks to solve other data-related challenges.

[1] A data mart is a subset of a larger database that is focused on a more specific domain or business need, allowing users of that data mart to extract insights within their domain of interest without navigating the complexity of the entire database. Querying a smaller, more bespoke subset of the database may also enable faster analysis because data volumes are generally lower.

[2] A FHIR server is a software system that can create, edit, delete, store, search, and transmit FHIR resources (e.g., FHIR representations of patients, providers, medications, and the links between them) in a way that follows a consistent, standardized specification defined by the HL7 organization. It typically does this by means of a RESTful API. For more information, see https://www.healthit.gov/sites/default/files/page/2021-04/FHIR%20API%20Fact%20Sheet.pdf and https://hl7.org/fhir/http.html.

OVERVIEW

THE PROBLEM

Public health data is messy, non-standardized, and siloed in multiple systems. At the same time, epidemiologists lack the tools and methods to efficiently turn it into meaningful intelligence that can drive timely public health action. Virginia experienced this problem throughout the pandemic and saw the prototype project as an opportunity to test paths and potential solutions to a more holistic approach to data processing and integration. Ultimately, the prototype project aimed to understand what could be done by starting from scratch and working with raw (i.e., unprocessed) data.

VIRGINIA'S CHALLENGE

The Virginia Department of Health wanted to improve their processes for making incoming data from health care partners analysis-ready to inform public health action. Specifically, they wanted to combine different data streams (ELR, or electronic lab reports; eCR, or electronic case reports; and VXU, or vaccinations) into a single database, where they could then quickly and easily perform analyses with minimal manual effort. Additionally, local jurisdictions—namely, Fairfax County—wanted to geocode the data to identify gaps in vaccination and then perform targeted outreach, such as holding vaccination drives within schools or apartment complexes.

GOALS

1 / **Explore new approaches to storing, processing, and linking different incoming data streams to yield robust, enriched, analysis-ready data and actionable insights.**

2 / **Ensure data is standardized, allowing for more efficient processing and more accurate analysis, and enable data to be exchanged with external partners in the future with minimal need for translation.**
For example, conforming data to the FHIR standard, which is already required and used for electronic health records (EHRs), could allow public health agencies and hospitals to share data bidirectionally.

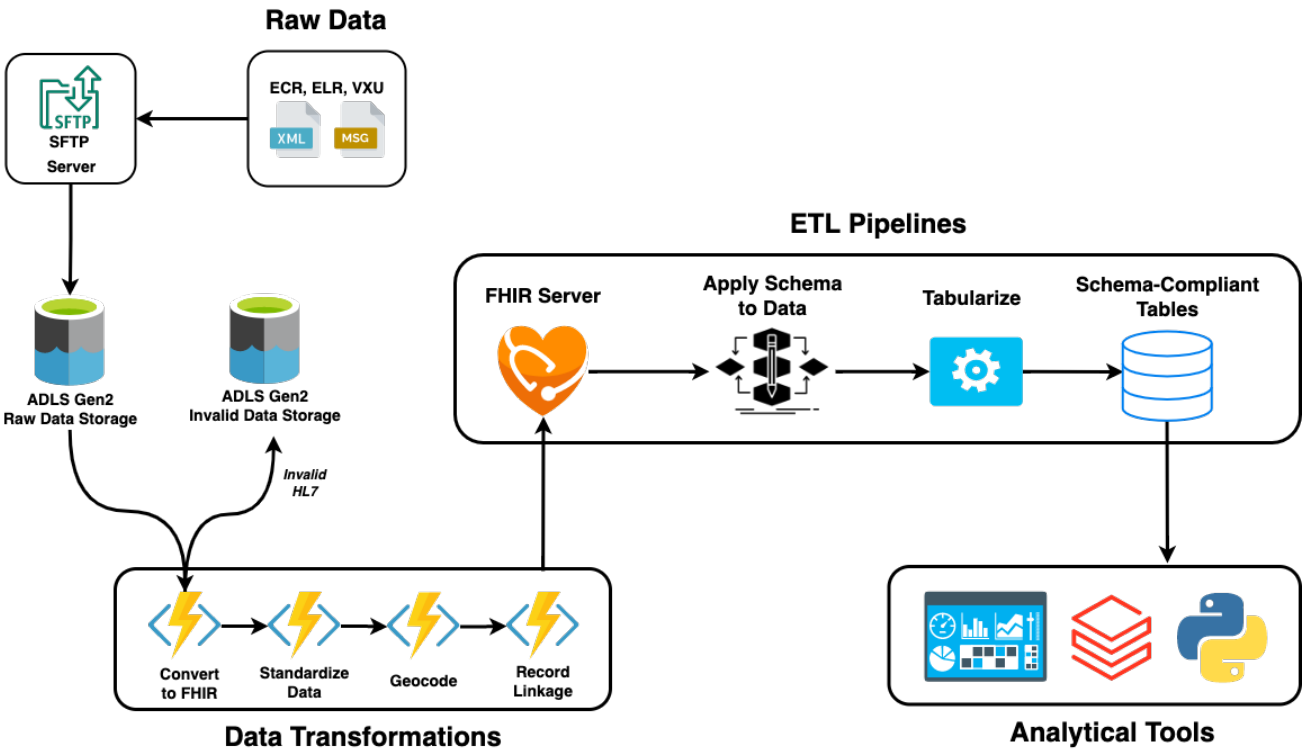
3 / **Test how reusable, modular tools (a.k.a. "Building Blocks"), such as a geocoding service, could aid in more targeted and informed public health action.**
Such a trial will also provide insight on creating a scalable and sustainable approach to building public health data tools, that minimizes additional burden on public health informaticists, minimizes one-off solutions, and creates off-the-shelf solutions for public health.

We have created a cloud-based, off-the-shelf data ingestion system where raw data sets (vaccines, case reports, and lab results) are processed in a single place. Data is standardized, deduplicated, geocoded, linked, and patient-level records are created to use for analysis.

ARCHITECTURAL DECISIONS

OVERVIEW

The Virginia pilot demonstrated that a single, streamlined process could be used within a public health department to validate, ingest, and link data across ELR, eCR, and VXU message streams. The prototype uses a cloud-based architecture, hosted in Microsoft Azure, to modularize[3] the processing of data from a variety of sources. For each step in the process, from ingestion to the creation of tables ready for analytics, a single function is applied to all eCR, ELR, and VXU data[4], removing the need for bespoke solutions for each data stream and creating a single source of truth for how to perform critical operations. The pipeline consists of several key architectural decisions, which we describe below:



The Virginia Prototype Synthesized Architecture Diagram

[3] Modularization is the process of converting a system or pipeline into discrete sections—each with a distinct function or purpose—for easier, more flexible construction.

[4] Within each step, a single Python function is used across all data streams, but each step in the data processing pipeline has a distinct function.

AZURE CLOUD ENVIRONMENT

The Virginia prototype is hosted in CDC's Microsoft Azure cloud environment, which was already supported at CDC and had many built-in tools to use during the prototyping. CDC was already operating an Azure environment, with staff in place to provide support on issues such as role-based access and standing up resources (e.g., Azure Data Factory) which simplified its adoption for this project. ***Based on what has been learned thus far, the use of Azure itself does not appear to provide any significant benefits over any other cloud provider.*** However, hosting this prototype in a modern cloud environment created significantly less work and less overhead than building a system on-premise. Best practice for data infrastructure includes the use of the cloud which provides many additional benefits, including horizontal scalability[5], serverless functionality, and cost-effectiveness, while ensuring the pipeline is fast, reliable, and dynamically scalable. Further, it provides metrics that are automatically tracked for any running resources, allowing for real-time insights into performance at all stages in the data processing pipeline.

PYTHON

The choice to use Python as the primary language for the Building Blocks was primarily driven by both its prevalence in the coding community (according to Stack Overflow, Python was one of the most popular programming languages in 2021) and in existing data tools (dbt, Airflow, and many other popular data tools are written in Python), as well as its ease of both reading and writing. Early interviews with Fairfax county and Virginia determined that this was a language that some epidemiologists used for daily operations, as well. Because the Building Blocks are intended to be the foundation of a community-driven ecosystem, it's imperative that the code used to define them is easy to understand and extend and requires minimal cognitive effort to integrate into existing tooling. In addition, Python provides seamless integration with all three major cloud providers and containerization, allowing the Building Blocks to be cloud-agnostic.

MODULAR ARCHITECTURE

Developing each of our data transformation processes as independent Building Blocks yields the following benefits:

- **Flexibility:** Creating a data processing pipeline from independent Building Blocks allows us to easily plug-and-play these operations together in different ways.

- **Ease of integration:** A modular paradigm provides the added benefit of creating tools and services that other jurisdictions can easily plug into their existing data pipelines, without requiring a complete re-architecture or overhaul of their systems. It also allows other jurisdictions to pick and choose which processes to integrate into their pipelines in the future, so they can avoid replacing parts of their ingestion process they do not wish to change and avoid adding in transformations they do not need.

- **Ease of updating:** VDH and other health departments can update, replace, or iterate on these individual data transformation processes in the future without having to upgrade their entire data ingestion system. We believe this will allow each individual process to stay more up-to-date, since the overall maintenance burden will be lower in the future, so they can avoid replacing parts of their ingestion process they do not wish to change and avoid adding in transformations they do not need.

- **Consistency of critical operations:** VDH's original data pipeline involved various forms of geocoding that all had slightly different business rules, and often created data inconsistencies. Performing critical data processing operations—like geocoding—using a smaller set of reusable, standardized functions ensures that there is a single source of truth for how key data transformations are implemented throughout a STLT's data ecosystem.

ISOLATING RAW, UNPROCESSED DATA

Connecting these Building Blocks to raw data streams "as-is" (i.e., upstream data that has not yet been transformed, restructured, enriched, filtered, or otherwise changed in format or substance by a surveillance system or other platform) ensures that these Building Blocks:

- Are more extensible to a broad range of jurisdictions
- Provide a single source of truth for how sensitive data processing operations (e.g., geocoding or standardization) are performed across data streams and data senders by minimizing upstream pre-operations on this data
- Can leverage the full breadth of information in the original unmodified records—in this case, ELR, eCR, and VXU data—to derive insight and action for public health practitioners. The analytical flexibility and depth of information available in raw "as-is" data are core value-adds of a data lake architecture, which is the traditional paradigm for storing various types of raw data in a unified environment. Specifically, raw "as-is" data is considered so rich because it is has not yet been subjected to any transformations that would result in data loss or a decrease in data integrity; raw data is therefore more valuable for as-yet-unknown analyses or investigations
- Can leverage modern blob storage[6] systems—such as Azure Blob Storage, Amazon S3, or Google Cloud Storage—as their upstream data source

FHIR SERVER

Our decisions to explore FHIR as a *lingua franca* for our Building Blocks, and to use a FHIR server, are central to the learnings of this prototype, and therefore deserve a separate, devoted discussion. See "An Assessment of FHIR" in the "Findings" section below for more information.

PARQUET

Parquet is a file format that is considered binary and columnar. In practice this means that the data is cheaper to store than it would be in other formats (e.g., CSV) due to high compression rates, and users can more efficiently run many data analysis queries because the data is stored by column instead of by row. This has significant advantages in analytics as most queries are designed to extract all row values for a handful of columns, as opposed to all column values for a handful of rows, meaning a file type that stores data by column—instead of by row—will read the data faster than other data formats (e.g., CSV). Parquet is also a widely-recognized data format, with integrations built in R, Python, and PowerBI. Additionally, it can be quickly converted to CSV format, which would allow integrations with tools like Tableau as well as most databases, particularly relational, SQL-based databases. Lastly, Parquet is a lightweight solution that doesn't require maintaining a database or its indices, it fits well with the modern, cloud-based architecture used by the rest of the pipeline. However, while the Virginia prototype uses Parquet files, future iterations of the Building Blocks will allow for persisting the data in cloud-based databases (e.g., Google's BigQuery, AWS's Redshift, or Azure's Synapse), other file formats (e.g., CSV), and local databases maintained by public health departments.

[5] Horizontal scalability is the concept of adding more computers to a system in order to make a process more performant. It is contrasted with vertical scalability, which is the concept of adding more resources (RAM, CPU, storage, etc) to existing computers in the system to improve performance. Horizontally scalable systems tend to outperform vertically scalable systems by parallelizing large operations and are much more easily scaled on-the-fly, e.g., scaled up automatically in response to spikes in incoming data volume. Horizontally scalable systems are typically more cost-effective as they allow system owners to pay only for computational resources that are needed at any given moment, instead of paying for machine resources that stand idle for large spans of time.

[6] Blob storage systems are data stores designed to store a wide array of unstructured data, ranging from text, to binary data, to CSVs, to HL7 or FHIR messages, to images and videos, and more.

INCOMING DATA

The prototype dealt with 18-days' worth of COVID-19-specific data from lab results, electronic case reports, and immunizations. Because current processes at VDH allowed them to store only 18-days' worth of data in their file transfer protocol (FTP) system, and because this prototype worked with a single snapshot of the data in that FTP system, the first six months of the prototype worked on data from an 18-day period. The data streams were in multiple formats; ELR and VXU were mainly in HL7 version 2 and version 3, while eCR were in XML (CDA). These data sets sent to VDH were often incomplete—missing important elements like race, ethnicity, or patient contact information—and included a high number of duplicate messages.

The variety of different data formats that data came to Virginia outside of the prototype work required business rules and processes in multiple siloed systems in order to take action on the results. The prototype instead ingested all three raw data streams, converted all of these data to a single standard format (in this case, FHIR), and standardized individual data elements to enable other downstream processes.

The team converted these data to FHIR to process the data in a single pipeline and increase record-processing speed. Additionally, by the end of 2022, all certified EHR vendors will be required to be able to exchange data using FHIR. In the future, health care organizations may prefer to exchange data with public health agencies using the FHIR standard. We hoped to understand how FHIR could be implemented for public health data, including both the challenges and potential improvements that come from the use of this standard.

After converting to FHIR, specific data standardizations, transformations, and enrichments were implemented to allow for deduplication and linkage of records:

- **Name and phone number** were standardized based on the current standard used by Virginia

  In the future, if this tool is iterated on and improved, data elements could be transformed to existing national standards, where they exist.

- **International names and non-English characters** were standardized and country codes were accounted for in phone number

- **Timestamp normalization** filled in missing Completion Status fields in VXU messages

  This improves the ability to successfully convert HL7 messages to FHIR. It allows data to be included into the rest of the pipeline and improves completeness of the data.

- **Address** was standardized to the United States Postal Service (USPS) standard

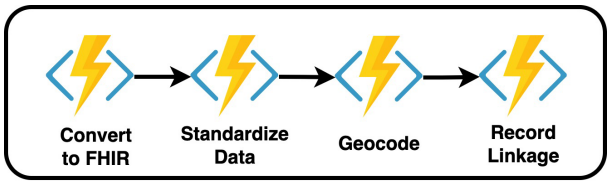
- **Geocoding** added latitude and longitude, and created an opportunity for spatial analysis capabilities and/or census block information

- **Linkage ID** combines records for a single patient across record types and also aggregates unique data across single records and deduplicates any data that exists across records

  This improves data quality, offering a more robust and comprehensive picture of a patient's health and identity. It is particularly useful when analyzing health equity themes across race and ethnicity since only select data streams typically contain this information.

BUILDING BLOCKS

The team determined the most effective way to process the raw data was to develop several modular tools, discussed above, that could be used consecutively to produce analysis-ready data. Given the use-case for the prototype (i.e., the identification of breakthrough COVID-19 cases) the team determined that the Building Blocks would need to convert all data to a single standard, deduplicate and link, and then geocode the cleaned results. The set of modular Building Blocks developed for Virginia are 1) Convert to FHIR; 2) Standardize Data; 3) Geocode; and, 4) Record Linkage. The team is working on an additional Building Block to create a data mart (i.e., a smaller domain-specific dataset extracted from the larger database in the FHIR server) for case reporting.



Building Blocks for Data Transformation

Longer term, several of these Building Blocks could have multiple solutions or approaches to achieve the same outcome. For example, there could be multiple options for Building Blocks that perform a similar function (e.g., geocoding or linking), but are configurable based on the STLT agency's needs, systems, and resources. Such decisions will be made in concert between CDC and STLTs, and informed by ongoing research.

While these Building Blocks were developed based on Virginia's needs, and on the COVID-19 breakthrough use case, their modularity and flexibility allow for widespread application. Through feedback and initial research with other STLTs, the team received feedback that standardization, deduplication, linkage, and geocoding services were high-priority needs, and could be used in a variety of public health use cases. In a later section, this paper details recommendations related to how to productionize and productize these services for use across STLTs.

While the Building Blocks were run on a limited amount of data in this prototype, the findings were significant and deserve further investigation on larger datasets.

FINDINGS

BENEFITS OF BUILDING BLOCKS PIPELINE

	FEATURE	ARCHITECTURE COMPONENT	VALUE
ORCHESTRATION	Real-time processing with automated retries	Event triggers, e.g., Azure Blob triggers, which instantly kick off a custom-specified process when a new payload is received	Data is processed as it is received, with immediate error handling and automated retries
RELIABILITY	System available 24/7	Managed cloud services, e.g., Azure Functions, Azure API for FHIR (Fully-managed FHIR Server); Zone-Redundant Storage, which stores data in multiple far-apart physical locations—both features guarantee 99.9%+ uptime	Data is backed up by infrastructure supported by large-scale cloud vendors; infrastructure availability is resilient to the failure of individual servers or entire data centers
SCALABILITY	Scalable infrastructure	Serverless cloud functions, e.g., Azure Functions, which offer baked-in support for on-demand horizontal scaling (scaling the number of machines that power a service to match data volumes), including "scale-to-zero" functionality when system is idle	Infrastructure is instantly auto-scaled up or down to handle the current workload to prevent bottlenecks, crashes, and unnecessary infrastructure costs
EASE OF USE	Single data pipeline for all data streams	FHIR Converter, e.g., Azure API for FHIR's conversion endpoint, which converts all HL7v2 and C-CDA payloads to FHIR R4; FHIR API (standard API for FHIR Servers), which offers versatile endpoints for create-read-update-delete operations on FHIR data, paired with a Building Block that creates custom tables out of data from a FHIR API; entire data pipeline built from reusable components	Removes the need for bespoke processes for separate data streams, allowing for a single pipeline to be used, debugged, and updated
SPEED	20,000 messages/hr processed end-to-end with potential for faster speeds	Horizontal scaling to process messages in parallel instead of sequentially, with computational power scaled to match data volumes on demand	Data volume does not serve as a bottleneck for STLTs, even when handling data spikes at COVID-volume
SYSTEM MONITORING	Ability to track live system performance	Performance monitoring services, offered by default by all major cloud vendors	Offers real-time visibility into what pieces of the pipeline are failing/are a bottleneck to the general health of the data workflow

SINGLE DATA PIPELINE

In Virginia's current state, incoming data is processed through their Rhapsody engine and then routed to distinct systems. Combining and reconciling lab and case data within the surveillance system relies on mainly manual processes, and adding vaccine data to determine breakthrough cases involves a large number of manual, time-intensive processes—as well as adjustments when additional doses are added to CDC recommendations. The high volume and spikes throughout the pandemic slowed down data processing as well, requiring complicated, scheduled runs that often took multiple hours to complete.

The architecture in the prototype funnels raw data into a single storage location—where it can always be referenced if needed—before data transformation occurs. Any data that errored out during transformation goes into a different storage location for invalid data, where errors can be addressed asynchronously. This approach increased the speed at which data was processed, allowed for transformations and other rules to be applied a single time—establishing consistency in how critical data operations are performed—and is scalable. Additionally, performance reporting is built into the architecture, allowing users to quickly notice issues, track metrics longitudinally, and make necessary adjustments.

AN ASSESSMENT OF FHIR

FHIR, and the systems associated with it—most notably the FHIR Server—were core technical components evaluated in the prototype process. In both the initial decision to explore FHIR, and in findings following the prototype, we have made the following observations:

BENEFITS OF CONVERTING TO FHIR AS A SINGLE DATA FORMAT

The conversion of the data from its original HL7 format to FHIR was motivated by two primary drivers. First, converting two different data types—pipe-delimited HL7 v2 and XML HL7 v3—into a single format[7] reduced both the number of processes that needed to be built, and the methods by which the data were accessed. This created a **single pipeline** that can handle a variety of incoming data formats. A bespoke solution would also require additional processes and Building Blocks to maintain depending on the incoming data.

Second, as mentioned earlier, the particular choice of FHIR as a data format was driven by upcoming regulations for EHRs that will require them to be able to exchange data using FHIR-based APIs by the end of 2022. As EHRs become more accustomed to exchanging data using FHIR, the preferred mechanism for sending data to public health could shift to FHIR over the next several years, and thus, in an effort to provide some level of future-proofing to the solution, the team implemented the FHIR standard as the default data format. There is also the added benefit of now having the data in JSON format (the data format used to represent FHIR data), which provides several advantages over other formats, such as those used by HL7 v2 and HL7 v3.

To help illustrate this point, the following lists highlight some of those advantages:

THE USE OF THE FHIR STANDARD PROVIDES:

- An active community of contributors, improving its capabilities, and building and maintaining tools that work with the standard
- A rich, comprehensive data model with broad coverage across the health data domain
- Capability to extend and define new elements to support never-before-implemented public health use cases
- Support for several key public health use cases such as electronic case reporting (eCR) and social determinants of health (SDOH)
- Versioning and history capabilities
- Search and join capabilities to support common analytics needs
- Bulk-export functionality to streamline ETL (extract, transform and load) processes

[7] This particular benefit comes from converting all data into a single format, and is not specifically unique to FHIR.

There are additional technical advantages that come with the use of FHIR as well:

- **Data access:** Compared to HL7 v2, which uses a series of delimiters (e.g., |, &, and ^) to differentiate the various fields and components of a segment, FHIR maps everything to key-value pairs, which are represented as "key": "value" in the data. This provides an easy means for accessing the data as each key must be unique, meaning the data can be accessed by looking for a specific key instead of splitting a segment by its delimiter and then counting the fields to look for a specific one.
- **Readability:** In initial discovery with Virginia and its respective localities, a common concern was the challenge around reading XML for eCR. Compared to the XML format used by eCR records, JSON data is considered much easier for humans to read as it reduces the amount of text needed to represent a data point. To demonstrate this, consider the following hypothetical representations of a patient's name:

  XML

  ```
  <Patient> <Name><Family>Smith</Family><Given>John></Given></Name></Patient>
  ```

  JSON

  ```
  {"Patient": {  "Name": {  "Family": Smith, "Given": John } }
  ```

- **Data exchange:** While the XML format used by eCR records is an acceptable format for data exchange, e.g., transmission via API calls, the JSON format is the most widely used format in modern API implementations, and thus lends itself better to more modern data architectures. This could allow STLTs to use commonly available APIs—such as those used by Apple Health or other widely available applications—to integrate additional data sources, assess integrated data, and perform outreach, with a reduced burden for data translation. Exchanging JSON is also faster and cheaper than XML, because JSON uses fewer characters to represent the same data, which is particularly important in cloud environments that charge users for storage space and network transfers (i.e., the volume of data sent "over the wire").

When processing the Virginia data and converting the messages to FHIR, ELR messages were valid (e.g., successfully converted to FHIR with no additional changes needed) approximately 14% of the time, and eCR messages were valid at a rate of roughly 67%. The primary driver of conversion failure was malformed timestamps in the original ELR messages. We therefore see these conversion failures not as a limitation of FHIR, but rather a reflection of the messiness of incoming data and the limitations of the particular FHIR converter we used, discussed further in the "FHIR Server" section below. As part of additional work throughout the month of June, the team built a separate upstream tool to standardize timestamps, in order to eliminate one of the most common reasons for invalid messages.

Several FHIR conversion solutions exist on the market today, including Azure HealthCare APIs and Linux for Health. General purpose transformation engines, such as Orion Rhapsody or InterSystems Iris for Health, may also be used to convert from various formats into FHIR. They generally provide a high degree of flexibility around how transformation is done. While an analysis of FHIR conversion alternatives was out-of-scope for this pilot, differentiating features among conversion solution providers include:

- **Cloud hosted managed solution vs. downloadable application solution**
- **Bulk conversion vs. message-by-message conversion**
- **Transformation definition language used to perform conversion**
- **Ability to define additional message types or adjust existing transformation definitions**
- **Scope of supported formats and message types for input files**

We also gave consideration to the alternative of unifying incoming data streams under a separate standard besides HL7 or FHIR, but that option surfaced a number of roadblocks. First, to our knowledge, no comparable and similarly robust standard for health data exists with widespread adoption and momentum behind it. Second, if we were to choose another standard (should it exist), we would have to port incoming HL7 and FHIR data over to that standard—presumably without standardized translation rules—which introduces additional complexity and a higher probability of data loss. Third, should we choose another format, we would likely lose the benefits of leveraging JSON, explained above.

DRAWBACKS OF FHIR

FHIR is a newer standard for health care and does not include all of the representations that public health agencies may need, like environmental or climate data. However, to our knowledge, no other health data standards include these data domains either, and there are early industry efforts to assess if and how environmental data and associated datasets could be adequately represented with HL7 or FHIR resources. FHIR resources, as currently defined by the HL7 organization, do encompass the majority of the data incoming to STLTs and will allow STLTs to exchange data bi-directionally—and with minimal implementation lift—with health care organizations and other government agencies (including other STLTs, or federal partners, like HHS or CDC).

Another notable drawback of FHIR is its suitability for direct analytics. Because of its nested structure and heavy use of object references[8], direct queries on FHIR data (in particular, queries analyzing various resource types) are noticeably slower than queries on data that is stored in relational (i.e., SQL-friendly) or similar formats. For that reason, our pipeline does not expose FHIR data directly for querying by an end-user. Rather, our pipeline involves user-driven curation of tabular data marts—extracted from the FHIR server—which are then stored in Parquet, a format that is much more performant for analytics.

[8] Object references entail representing a linked object (e.g., a patient's provider) by an ID that refers to the original copy of that object which is stored elsewhere in the database. Object references can be contrasted with storing a complete duplicate copy of a linked object in the place where that linked object would otherwise be referenced via ID.

FHIR SERVER

Beyond the use of FHIR as a *lingua franca* for our Building Blocks, the FHIR server — located at the center of our Virginia prototype architecture—provides numerous benefits:

- **Conversion:** Converting the data to a standardized format—which, for the Virginia prototype, was FHIR—makes it possible to provide a single approach to processing the three data streams. The FHIR Server is the tool by which this conversion is done. For this prototype, we used Azure's fully managed FHIR server, Azure API for FHIR. Other FHIR conversion platforms exist but conducting an analysis of alternatives for FHIR conversion was not considered in-scope for this pilot and is a task for future research.
- **Validation:** As part of the conversion process, the FHIR Server also automatically checks if the incoming HL7 and CCDA messages are valid according to the templates put in place by Microsoft.
- **Single source of truth:** Once the data has been processed, it is stored in the FHIR Server, providing a single source of truth from which all analytical tables can be built. This ensures that users are getting the benefits of the data processing, and any given table reflects the same data as any other table.
- **Data exchange:** While not leveraged for the Virginia pilot, the FHIR Server provides a robust API with extensive querying capabilities that, with the proper authentication and authorization protocols in place, provide a seamless mechanism for exchanging data across jurisdictions and for use by applications. In the near future, we aim to expand upon this prototype to test the ability of our reference architecture—namely the FHIR server—to support more seamless interjurisdictional exchange alongside a data ingestion workflow.

In addition to the numerous benefits provided by the FHIR Server, the choice to include it in the architecture was also driven by the direction the industry is headed, as mentioned above. As regulations mandate that EHRs implement mechanisms by which to send and receive data in FHIR format, a future where data in the FHIR format becomes more common is a likelihood, meaning that public health departments that have a FHIR server in place will be ready to also receive that data.

In discussing the benefits and drawbacks of a FHIR server, it is important to consider the feasibility of migrating data from a FHIR server to another SQL or NoSQL database. Below, we offer a brief discussion assessing migration to various alternative data stores:

- **NoSQL databases** such as Athena, MongoDB, DynamoDB, CosmosDB, ElasticSearch, and many others offer a relatively low burden for migration, primarily because these databases would allow migrated FHIR data to be stored in its native JSON format. However, each of these other NoSQL options would pose their own complications, including setting up clusters, learning the query language to extract the data, and defining collections.
- **FHIR data** can also be migrated to a relational database. That process would require explicitly mapping the entity types and fields from JSON-based FHIR data to designated tables and columns under a relational paradigm, which entails a heavier migration lift relative to NoSQL migrations, yet is still quite manageable. One may also wish to migrate FHIR server data to FHIR-specific RDBMS[9] solutions like fhirbase (which rely on Postgres under the hood), but those solutions are closed-source and proprietary, and therefore may present additional migration obstacles.

[9] Relational Database Management System

CASE REPORTS

Case reporting, or condition reporting, is mandated by state policy for clinicians, laboratories, and other entities to report the diagnosis of a determined list of conditions (e.g., COVID-19, tuberculosis, sexually transmitted infections (STIs), Lyme disease, and others) to jurisdictional public health agencies. The way these data are reported to STLTs is dependent on the reporter and the state, and ranges from manual (e.g., phone calls, faxed or mailed paper reports, web entry) to automated electronic messaging.

Electronic case reporting can help fill the gaps left via manual reporting mechanisms. In Virginia, six health care organizations are sending electronic case reports for COVID-19 through the use of an intermediary: the Association of Public Health Laboratories' (APHL) Informatics Messaging Services (AIMS) platform. Through this intermediary, health organizations electronically send all of their case reports to this single system, where it is processed through a series of rules (termed the Reportable Conditions Knowledge Management System, or RCKMS) and then sent to appropriate public health agencies. RCKMS is a proprietary system integral to the AIMS platform that STLTs are able to update based on state reporting requirements. The reports are sent following a specification called eICR: the electronic Initial Case Report, developed through HL7 in collaboration with APHL and other public health associations.

THE TOP CHALLENGES FACED BY VIRGINIA ARE SUMMARIZED BELOW:

- Issues and resource constraints with validating data from hospitals and health care organizations already onboarded through AIMS
- The volume of information included in the case report, including old or irrelevant clinical information
- The data that is parsed is limited due to constraints in the instance of Virginia's case surveillance system, making the review and use of it more manual and time-intensive
- Any electronic updates to the case report results in a new message, rather than updating the existing report
- Lack of specific provenance (contextual) data, making it challenging to understand where and when certain data was entered
- No standard case reporting data mart exists natively in the system because limited data is parsed making analysis manual and time-intensive
- Phone calls and follow-ups with providers and patients are still required to complete data requested in case investigation forms in their surveillance system

Many of the above challenges experienced by Virginia are related to the volume of information included in the case reports, and the challenge of parsing the components to feed into their surveillance and reporting systems. Rather than targeted reports based on condition, they are a "firehose of data," many times including all the patient's active diagnoses in the medical record, medications administered, and other clinical information that can sometimes span the patient's entire medical history, when instead more limited clinical data relevant to the reportable diagnosis and the associated encounter would be most immediately helpful for informed public health action. Virginia has received feedback from its local jurisdictions that the reports in their current form are "burdensome," and it takes significant manual time and effort to comb through the data in order to find actionable information. This challenge is exacerbated by the format and structure of the incoming reports (XML), and the lack of parsing of the data, which is a limitation of the version of VDH's surveillance system.

Epidemiologists at VDH have expressed particular concern about the volume of data included in case reports. They have expressed concerns about health status information being included in case reports for conditions where the data is not relevant to investigation or treatment.

Some of the challenges with the volume of data can be attributed to upstream issues, such as problem list management, the way in which symptoms and history of present illness are documented in EHRs (e.g., free text versus codes), and the lack of associated context about data elements (e.g., difficulty in discerning which provider or facility entered the clinical data or oversaw the relevant clinical visit or event). However, tools that filter only the most relevant and associated data could help in the interim, before solutions for upstream data management and exchange are implemented. A longer-term solution could include querying of EHRs based on a trigger—such as a lab result, or a hospital admission—or having read-only, audit access to an EHR to replace manual outreach to hospitals, clinical support staff, and patients.

IN THIS SCENARIO:

- **A public health official would receive a lab or a diagnosis for a reportable condition that included patient demographic information**
- **The system would then search for the patient using that information (name, date of birth, phone number, address, or a combination of these) and pull needed additional relevant information:**
    - **Within the health care organization's EHR—if direct, read-only access was available**
    - **Within a regional or national health information exchange (HIE) that contained patient-level clinical information for many clinical facilities**
    - **By querying via an API that could pull data from multiple facilities' EHRs**

This would allow public health staff to search for targeted, relevant information within the source, and save time and resources from parsing through large, unparsed reports. This is a reality today in some local health agencies, and more research and work to understand the pros and cons of this approach are needed. Particularly, these queries and methods of searching should be as automated as possible, to ensure these mechanisms are not placing new or undue burden on public health officials. As new approaches are piloted, it will be important to measure time and resources spent to ensure STLTs adopt the tool(s) that are most efficient and effective for their incoming data and associated workflows.

After health care organizations complete their implementation with AIMS, Virginia needs to complete a series of complex, resource intensive steps in order to get the data into their surveillance system (Virginia Electronic Disease Surveillance System, VEDSS, which is an instance of CDC's NBS). These steps include making custom filters in VDH's Rhapsody engine to: 1) ensure data is only accepted from organizations that have been thoroughly tested and deemed compliant and production-ready for approved conditions; 2) create a copy of the original eCR for each reportable condition within it, per Virginia's reporting specification; and 3) route non-approved conditions by facility to VDH's test environment for further validation. Further steps for Virginia include creating and maintaining a custom, self-developed eCR data mart to allow epidemiologists to perform analyses; and importing that data into VEDSS. However, importing into VEDSS does not result in the data being parsed, limiting its usability for analysis.

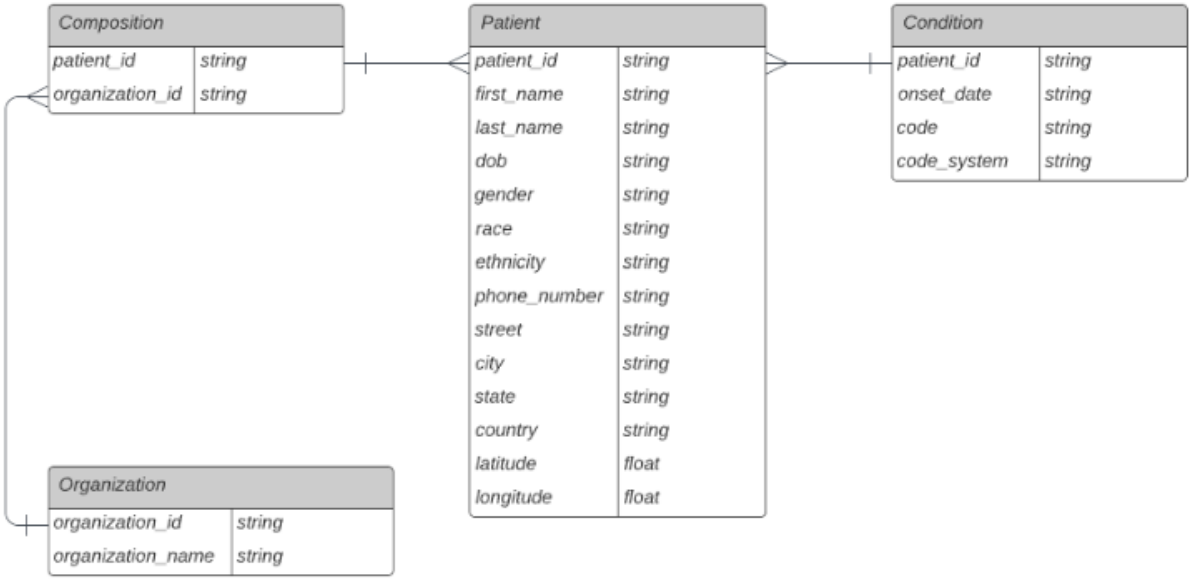
Further, Virginia has struggled with getting summary statistics or monitoring data from RCKMS[10], including volume from senders or completeness of messages. As a result, Virginia has developed an eCR data repository in SQL to improve QA for data completeness. Enhancements to that eCR data repository are continuous as new challenges arise.

[10] RCKMS is the initial receiver of the case report from the reporter (hospital or provider organization). Virginia is looking for summary data from that initial source to understand volume and completeness of incoming data. It is much more challenging, as those reports go through several processing steps before they get to VEDSS, for VDH to get monitoring data from their own systems (Rhapsody or VEDSS).

As mentioned briefly above, the desire for **targeted, condition-specific data** could be supported by triggered messages based on diagnosis or lab results, and access to the reporter's EHR, or a regional HIE, to query for relevant data (ideally, through a FHIR-based API). Another longer-term option to receive more targeted, specific data is to work with vendors and providers on upstream challenges, such as problem list maintenance, the use of national standards, and ensuring the specification is implemented appropriately. Working closely with the Office of the National Coordinator for Health Information Technology (ONC) on the development of data standards for public health is vital to ensure that those standards provide the needed data elements for public health action, can be easily parsed and analyzed within public health systems, and do not include irrelevant or out-of-date information.

It is particularly timely to address potential improvements in implementation and onboarding for eCR, as the Centers for Medicare and Medicaid Services (CMS) will require electronic case reporting to jurisdictional public health agencies in 2022 for hospitals and 2023 for providers. As such, states will have a substantial increase in health care organizations looking to onboard for electronic case reporting, and therefore an associated increase in case data. It is important that STLTs receive **relevant, actionable, and analysis-ready information,** and are not left to manual interventions and designing bespoke solutions in order to make use of these data. The use of the eCR-specific data mart could be an interim solution to allow users to view and analyze the most important data in the incoming reports, while still preserving the additional elements in the event they are useful for further investigation or intervention.

As a part of their prototype work with Virginia, the team created an eCR data mart through a web application that provides an interactive means of generating schemas. Schemas, an example of which is shown below, are used to define tables and show how those tables relate to each other. Using the web application, the team was able to define the eCR schema using a point-and-click interface, which was then automatically converted to FHIR API calls. Those API calls wereused to extract data from the FHIR server to create the tables represented in the schema. This approach of using a web interface to define the schema and programmatically convert that into a series of API calls removes the need for users to be intimately familiar with the FHIR API, allowing them to focus on the analytics and reporting. The next steps for a more robust version of this web application would be to ensure it supports joins, which would allow for tables to be populated with data from multiple resource types (including reports, dashboards, and SQL queries) and to connect with FHIR structural definition feeds so the application is automatically updated with fields from any given resource type, thereby enabling users to specify what data they want and for what conditions.



An example of the eCR-specific schema

LINKAGE

Record linkage, or patient matching, identifies multiple records referring to the same individual and combines them into a single, more complete, patient record. While patient matching is a persistent problem for providers and health care organizations, the challenges—and demand for a solution—extend to public health as well. In a public health context, linking records both within and across data streams provides more accurate statistics around patients and diseases. Furthermore, automated linkage also reduces the need for public health officials to manually associate records, a task that is currently a major burden to complete.

Virginia was able to do basic linkage in VEDSS among lab and condition reports along two tiers. VEDSS first tried to match incoming records using any available ID codes[11] that match existing records. However, some of those ID codes, like local registry ID, are not included in eCR records coming directly from providers (versus being imported directly from another NEDSS-based system), nor are they included in any ELR records at all. Nearly all other ID codes VEDSS uses for matching, such as CHIP ID, Medicare number, and SSN, are also rarely found in incoming records (this was not a unique limitation of VEDSS, but rather a challenge posed by the data that public health departments typically receive). Assuming no match using ID codes, VEDSS then tried to match patient records on the combination of legal name, date of birth (DOB), and sex.

Along each matching criteria tier, VEDSS needed an exact match, otherwise a match could not be made. The process continued until no tiers were left, at which point a new record was created. If VEDSS found only a partial match in any tier, e.g., a match on name and sex but not DOB, the incoming record was added to a long "possible match" queue that required human review to determine match or duplicate status.

Additionally, combining immunization records to identify COVID-19 breakthrough cases required a separate database with additional manual steps: patients in the queue had to be manually vetted, then combined in the patient database, then moved to a separate query database, then combined with vaccination data in which there were also possible duplicates, all before analysis was possible. The complexity of the task grew as changes in vaccine recommendations became changes in analytic criteria, such as the introduction of a booster dose moving the target for a breakthrough infection. This complexity added more duplicative data with compounding time complications.

The new linkage Building Block adheres to several best practices for patient matching:

- It bases matches on **at least three salient patient features**, e.g., name, DOB, and address.
- When used in concert with upstream Building Blocks, our linkage algorithm **standardizes patient data** (e.g., removing non-alphabetical characters from names) before using that data to link records together.
- It creates a **hashed linking identifier**, which condenses name, DOB, and address into a single non-human-readable field; while also facilitating and simplifying linkage, as records need only be compared by the hash, rather than the three fields individually.
    - Because hashing is not a reversible operation, this hash further **enhances and preserves privacy**, as it prevents extraction of protected health information.

[11] Local registry ID, account number, alternate person number, CHIP identification number, driver's license number, Medicaid number, medical record number, Medicare number, mother's identifier, national unique individual identifier, patient external identifier, patient internal identifier, person number, prison identification number, Ryan White identifier, SSN, visa/passport number, and WIC identifier.

The new linkage Building Block also made several improvements over the current state:

- It uses a **standardized, geocoded address,** in addition to name and DOB, to determine an exact match.
    - Research has demonstrated that the use of standardized data over non-standardized data in patient matching improves match rates by up to 3%, and standardized addresses are particularly valuable in this regard.[12]
- It uses patient attributes with **higher information content** to conduct its matches, which increases match accuracy.
    - The smaller the cohort of individuals that are uniquely characterized by an attribute's value, the more useful that attribute is for matching—this aspect of an attribute's value is known as its information content.
    - The information content of address information is greater, and therefore more useful for patient matching, than sex, because the number of people who live at "100 Main St, Pleasantville, NY" is orders of magnitude smaller than the number of people whose listed sex is "M".
    - Notably, the use of sex information for patient linkage is generally considered error-prone given the heterogeneity of standards and methods for how sex is recorded.[13]
- It linked both within and across the three data streams upfront, earlier in the data pipeline, creating single records to serve as the **source of truth** for downstream analytics. (Note: this prototype did not link new data with existing historical data processed by VDH).
    - Creating a source of truth at the beginning of the pipeline is a big win for Virginia: "it's the single biggest value statement…in moving data treatment upstream."[14]
- It added a **cryptographic salt** (random characters to obfuscate encoded information) on top of its hash codes, using a hidden key to protect anonymity. This further **enhanced patient privacy** by creating an additional barrier to prevent extraction of protected health information.

[12] Shaun J Grannis, Huiping Xu, Joshua R Vest, Suranga Kasthurirathne, Na Bo, Ben Moscovitch, Rita Torkzadeh, Josh Rising, "Evaluating the effect of data standardization and validation on patient matching accuracy," Journal of the American Medical Informatics Association, Volume 26, Issue 5, May 2019, pages 447—456, https://doi.org/10.1093/jamia/ocy191.

[13] Source: Bohensky MA, Jolley D, Sundararajan V, Evans S, Pilcher DV, Scott I, Brand CA. Data linkage: a powerful research tool with potential problems. BMC Health Serv Res. 2010 Dec 22;10:346. doi: 10.1186/1472-6963-10-346. PMID: 21176171; PMCID: PMC3271236.

[14] Tim Powell, May 4, 2022, Demo.

LINKAGE RESULTS

% REDUCTION IN # PATIENTS	ELR	ECR	VXU	COMBINED
	15%	29%	11%	19%

- For each individual stream, a significant number of records were linked, resulting in meaningful deduplication.
- The effect of linkage on deduplication across all combined data streams was additive upon the effect of linkage on the individual streams (~10k records). *This is a substantial and meaningful finding—it indicates that bringing together different sources of data enhances and improves completeness of these data.*
- Although the data sample was only several hundred thousand records over an 18-day time frame (a short window, which diminishes the likelihood of multiple records for a single patient and therefore diminishes the linkage effect), these initial findings are very encouraging and suggest the possibility of a more pronounced, stronger patient matching effect in a larger body of data and longer period of time.

Findings also indicated **demographic field recovery** when combining data from different sources. This means patient matching also supplemented or recovered fields that were missing one of the records, but was present in another. (For example, if Patient A and Patient C are linked matches and share a hash, and A's record was missing ethnicity but had gender, while B's record has ethnicity documented but not gender, the singular patient record now contained both gender and ethnicity). Despite an HHS rule that requires COVID ELR records to report race and ethnicity, these fields were not present in all incoming messages. As a result of the linkage Building Block, the completeness of race and ethnicity data improved. After linkage within ELR data, there was a 2% decrease in ELR records missing race, and a 3% decrease in ELR records missing ethnicity. Findings with cross-data stream linkage had lower rates of improvement—about 1%—likely due to the high volume of duplicates removed when combining these data.

While the benefits gained from this step are notable, there are multiple approaches to patient matching and record linkage beyond deterministic algorithms (i.e., the approach used in this Building Block), which look for exact matches on fields beyond ID codes like SSN, drivers license number, etc. Probabilistic matching uses an algorithm that determines "the likelihood that two records refer to the same individual even if typos or other irregularities exist in the data," and provides the likelihood of a match.[15] Use of these more advanced algorithms does require some level of human intervention to review "potential match" queues that do not meet the set thresholds for matching.

[15] Urahn SK, Rising J, Moscovitch B, Torkzadeh R., "Enhanced patient matching is critical to achieving full promise of digital health records," The Pew Charitable Trusts; 2018, https://www.pewtrusts.org/-/media/assets/2018/09/healthit_enhancedpatientmatching_report_final.pdf.

By contrast, referential matching is the process that "leverage{s] data from different sources to build a more complete profile of each patient that includes past addresses, common name spellings for individuals, and other demographic data that changes over time."[16] Referential matching uses common, non-health sources to supplement missing data, and requires the use of a third-party vendor due to the use of external, non-health care data sources, such as credit or other financial data. Referential matching can be supported by the use of a master person index (MPI), though not strictly necessary.

While all of these processes have pros and cons, recent research has found that referential matching has advantages when compared to probabilistic matching; a study found referential methods to have improved accuracy, both due to demographic data supplementation, and the associated expanded matching logic that additional data elements made possible.[17] However, depending on the needs and resources of the STLT, different approaches for linkage may make sense.

In the longer term, the team believes that this Building Block should offer multiple linkage options to meet the varying needs and resources of STLTs: basic deterministic, of which there is a current prototype; probabilistic; and, for more advanced users, referential, offered through shared licensing with a third-party vendor.

[16] Ibid.

[17] Shaun J Grannis, Jennifer L Williams, Suranga Kasthuri, Molly Murray, Huiping Xu, "Evaluation of real-world referential and probabilistic patient matching to advance patient identification strategy," Journal of the American Medical Informatics Association, 2022, https://doi.org/10.1093/jamia/ocac068.

NEXT STEPS AND RECOMMENDATIONS

BEYOND VIRGINIA

Virginia's partnership, feedback, and willingness to share data and collaborate on early iterations of the Building Blocks has been invaluable. The team would not have been able to understand the current and desired state, and design tools that solved real pain points, without their engagement, openness, and continued trust in the work. Their collaboration has not only been foundational to the creation of the initial Building Blocks themselves, but also for the higher-level design of an infrastructure and environment that can support the broad implementation, iteration, and support of modular tools at a national level.

Beyond Virginia, it is important that Building Blocks can be used by public health departments with diverse technical capabilities, regardless of vendor or infrastructure. Different departments receive data in different ways, from a variety of reporters, and have a range of systems and technical resources to ingest and analyze incoming data. Some public health authorities rely on basic tools, like Excel; others use on-premise servers; many are in the process of moving, or have already moved, to the cloud; others yet use multiple cloud vendors for different programs and datasets.

By using Building Blocks with additional data sources and use cases, public health departments help ensure that these tools solve problems beyond COVID-19 specific needs. It is important that Building Blocks are sustainable and can improve the data pipeline for public health needs today and in the future.

CDC and the public health community can endeavor to design Building Blocks in a way that makes it easy for STLTs to access and incorporate the tools into their production environments, across a variety of architectures. This could be done through the following:

- **Continually conducting user research for iterative human centered development**
- **Ensuring Building Blocks are always as modular as possible**
- **Maintaining consistency across SDK taxonomy as features evolve and grow**

Some near-term opportunities to pilot, research, and develop Building Blocks could include:

- **Completing research to understand the needs of STLTs across the technical maturity spectrum, including how Building Blocks can improve data pipelines, and inform infrastructure recommendations**
- **Standing up a marketplace to house the Building Blocks as they become production-ready; Designing Building Blocks to be easily portable across STLTs' systems with minimal lift**

This continued focus on the importance of STLT-driven participatory design models allows for the evolution of public health modernization, ensures tools are meeting the needs of the users for which they are designed, and that these tools can adapt as needs and demands change.

INFRASTRUCTURE CONSIDERATIONS

While some of the work from this prototype is specific to VDH and the infrastructure the team worked in (e.g., Azure), there are learnings that have informed larger architectural modernization strategy that can be applied to other STLTs that may be in the early stages of upgrading their infrastructure. It is important that STLTs take steps to modernize, as they are able, with support from CDC, CDC Foundation, and national associations.

In order to maximize the benefit of Building Blocks, STLTs should consider the following foundational infrastructure tenets as they begin to modernize:

- **Migration of data and systems to the cloud**
  Note: Cloud-based architectures offer support for cost-effective, dynamic horizontal scaling that automatically increases or decreases system capacity to meet incoming data volumes.

- **Performance monitoring for data ingestion pipeline visibility and troubleshooting**
  Note: Most cloud vendors offer extensive performance monitoring for little to no additional cost as part of their offerings.

- **Access to raw, unprocessed data that has not yet been ingested into surveillance systems, and maintaining long-term access to that data as allowed and feasible based on costs and policies**
  Note: Standard data lake architectures achieve this.

- **Use of at least one of the following analysis tools:**
  Open-source SQL-based relational database management systems, modern data science and engineering languages like R and Python, and modern data processing, querying, and visualization tools like Power BI, Azure Synapse, Google BigQuery, Amazon Redshift, or Tableau & Tableau Prep.

APPENDIX:
PRODUCT OFFERINGS AND SUPPORT

HOW TO CREATE AND OFFER PRODUCTS

There is a wide range of technical capacities among STLTs, and a singular product will not serve the diverse data workflows that currently exist among jurisdictions. The outline below presents potential product offerings that could support Building Blocks; such offerings range from a marketplace of Software as a Service (SaaS) solutions with shared "usage patterns" (e.g., SaaS solutions that are all APIs that obey a common specification) to Software Development Kits (SDKs) that are similarly structured to a full-fledged CDC-hosted product that lets STLTs store and/or serve data in a cloud environment. The sequencing of how and when these products become available largely depend on additional research around the needs and desires of STLTs.

Key questions include:

- Which Building Blocks would be **most valuable to transition into a product today**, for the greatest number of STLTs?
- Should **one STLT (or many)** be prioritized for piloting a Building Block in production, before iterating on it to meet the needs of other jurisdictions?
- How can Building Blocks be **accessible** to STLTs of varying technical capabilities while ensuring a unified and scalable approach for adoption?
- What is the best way to serve up/offer Building Blocks to **encourage adoption?**
- How might **feedback loops** be created to measure qualitative and quantitative improvements once a Building Block is adopted and used?
- What sort of metrics define **success for a Building Block?** (e.g., adoption rate, efficiency improvements)
- What is the best way to create a **collaborative, open-source community** to continually develop, support, and iterate on these Building Blocks?

PRODUCT OFFERINGS

SDK

An SDK offering would house all the logic related to each feature and make those features publicly available through language specific libraries. SDKs are a common practice to leverage features and functionality that individuals, companies, and organizations have already composed. There are library package managers for most well supported programming languages: pip for Python, CRAN for R, npm for Javascript, Maven for Java, and more.[18] The first SDK offering should be developed in a language that supports pilot partners, and then expanded to other languages over time as STLTs request additional language support.

For an SDK to be successful, there should be considerations around research-informed features, performance, documentation, scalability, security, and error handling.

We see examples of successful SDK products across a variety of sectors. A few examples are:

- Mapbox SDKs **for customized web maps, search, and navigation**
- Twilio SDKs **for customized communication services**
- TensorFlow SDKs **for machine learning services**
- tidyverse SDKs **for data science functions**

PUBLICLY AVAILABLE DOCKER FILES

Publicly available Docker containers would enable STLTs to mirror a microservice architecture utilized by other STLTs, an architecture that has been proposed by the CDC, or an architecture of their own creation, while minimizing variations in environmental dependencies that make microservices more brittle or inconsistent. Docker-compose files allow multiple containers, each of which encapsulates a single Building Block, to be orchestrated in a single configuration. By supporting containerization for the major cloud providers (e.g., AWS, Google, and Azure), there could be a standardized manner to test and deploy the Building Block data pipeline in a STLT's cloud environment in a cloud agnostic fashion. Another benefit of Docker is that both the production environment and the local environment are identical, enabling seamless development and experimentation, as well as scaffolding for STLTs to own and operate one or more data pipelines themselves.

DRAG AND DROP USER INTERFACE (UI)

Since STLTs have varying technical capacities, there could be multiple points of entry into the Building Blocks ecosystem. One potential offering for STLTs with less technical capacity is to create a Building Block portal where users can drag and drop Building Block tools into a data pipeline via a UI or drag reporting data into that pipeline via a UI. The data processing could go through conversion to FHIR, field standardization, and schema transformation processes, but the trigger to start the data processing would be a manual upload rather than an automated trigger managed by cloud infrastructure or web services.

[18] Ovidijus Okinskas, "What is a package manager and why should you use one?" IDR Solutions, July, 11, 2018, https://blog.idrsolutions.com/what-is-a-package-manager-and-why-should-you-use-one/.

APIS

An API product offering would allow STLTs to take advantage of CDC hosted web services. These web services could be designed differently depending on the needs and interests of the STLTs.

1 / **API for single data processing job *without* data storage.**
This would create multiple API-powered processing pipelines for data conversion and standardization. These API options offer features for processing/enhancing the data, but not hosting the data.

Examples include:

- *A CDC web service that ingests HL7 messages for eCR, eLR, VXU, ADT, and other incoming data streams and converts to FHIR, standardizes fields, and outputs FHIR bundles. STLTs could customize processing via a configuration file and would receive the output files in a cloud container that would expire after a predetermined period.*
- *A CDC web service that handles FHIR bundles and converts those bundles to a data mart, based on job configurations and a predefined schema.*
- *Taking in a FHIR bundle and outputting a file for the surveillance system (e.g., like the standard NBS input file)*

2 / **API for single data processing job *with* data storage.**
This would create multiple API-powered processing pipelines for data conversion and standardization. This option offers features for both processing/enhancing the data and hosting the data.

Examples include:

- *A CDC web service that ingests HL7 messages for eCR, eLR, VXU, ADT, and other incoming data streams and converts to FHIR, standardizes fields, and stores data in a FHIR database hosted by the CDC. STLTs could customize processing via a configuration file and then access their data using the pre-established FHIR endpoints for querying and exporting.*
- *A CDC web service that handles FHIR bundles and converts those bundles to a data mart, based on job configurations and a predefined schema. STLTs could access the data mart via CDC owned cloud services.*

3 / **API per Building Block.**
Multiple APIs exist for each piece of functionality in the Building Block pipeline. There would be individual endpoints for conversion from XML or HL7 to FHIR, name standardization, phone number standardization, geocoding, and any other future-developed Building Blocks. This option would offer processing/enhancing the data per Building Block but would not include hosting the data.

HOW TO SUPPORT PRODUCTS IN AN OPEN-SOURCE COMMUNITY

The code from these product offerings could live in an open-source repository for community contributions, maintenance, and feedback.

In order to facilitate a participatory approach to the code base, the following areas should be considered at the beginning stages of a project:

Governance
Who are the core contributors?
What permissions are available to those who aren't core contributors?

Documentation
Is there documentation about core functionality and how to run the application or library?

Maintenance
Who will maintain the project?
Who is responsible for user support?

Licensing
What kind of license does the code have?
What kind(s) of license(s) are needed for third-party services used in Building Blocks?

Contribution standards
What are the guides for the project in terms of code design, test coverage, and more?
Is there a code of conduct? A changelog?

Sustainability
It is important that support—financial, human, and technical—for these products are long-term, and will not disappear from funding priorities.

PRODUCT RECOMMENDATIONS

Moving forward, we recommend the following product offerings:

PYTHON SDK

Having the first product offering as an SDK helps modularize the logic behind each Building Block and makes it publicly available as a library so that STLTs can import, experiment, and integrate these features into their existing pipelines. Centralizing Building Block efforts to the repository allows partners to potentially contribute, upgrade, and/or add to existing functionality. An SDK is the first step to cultivating an open-source community around the data modernization initiative for public health.

PUBLICLY AVAILABLE DOCKER FILES FOR CLOUD-BASED MICROSERVICES

Publicly available Docker files for cloud-based microservices allow STLTs to take the next step with the Building Blocks pipeline—standing up their own cloud-based infrastructure for streamlining data sources, standardizing data fields, and converting to FHIR. Since Docker enables replicas of production environments, there is little ambiguity around what the production microservice offers, but a lot of flexibility for customized features and configurations.

APIS

An API for Building Blocks would be a powerful tool for STLTs to utilize as part of their data processing workflows. There are several ways to offer an API for the data pipeline product. Deciding on a particular API interface will be largely informed by research about STLTs' willingness to:

- **Send data through CDC data systems**
- **Host data in CDC data systems**

An API is also a significant pivot in product offerings, because Phases 1 and 2 enable STLTs to incorporate Building Block tooling into their own infrastructure ecosystem. An API creates a significant shift of responsibility onto the CDC to make decisions about short- and long-term data storage, when processing, enriching or serving data. Having this type of product would require a long-term plan for product maintenance and user support.

DRAG-AND-DROP UI

APIs will likely power any type of drag-and-drop UI, so it's necessary to build the backend scaffolding around the pipeline access points before considering a visual interface. Once APIs are in place, a portal for STLT partners can be developed. There is still a lot of research needed to inform how much value a UI for this pipeline would offer.