Project Lantern Data Sources and Linking Mechanisms

Sponsor: Assistant Secretary for Technology Policy/Office of the National Coordinator for Health Information Technology (hereafter ASTP)
Contract No.: 47QTCB21D0023
Project No.: 100905.10.100.1000.AA0

Columbia, MD

Authors:
Matt Mayer
Emily Michaud
Brianna Mathiowetz
Prasad Konka (SVG, Inc.)

May 2025

Executive Summary

Lantern is an open-source tool developed by the Assistant Secretary for Technology Policy/Office of the National Coordinator for Health Information Technology (hereafter ASTP) and Mettle Solutions, LLC that monitors and provides analytics about the availability and adoption of FHIR API service base URLs (endpoints) across healthcare organizations in the United States. It also gathers information about FHIR Capability Statements returned by these endpoints and provides visualizations to show FHIR adoption and patient data availability. Lantern sources most of its data from publicly available endpoint and organization lists, though some of the data is generated from the Lantern application itself. This document details the publicly available data sources and explains the processes used to produce data by the Lantern application.

Table of Contents

List of Tables

1 Endpoint Data

The Lantern project uses publicly available endpoint lists to generate an aggregated list of FHIR API endpoints. The majority of lists now come from the Certified Health IT Product List (CHPL) by querying the **/search** API for g(10) certified products. Outside of CHPL, there are a few publicly available lists that Lantern has included:

- CareEvolution (https://fhir.docs.careevolution.com/overview/public_endpoints.html)
- 1upHealth (https://1up.health/fhir-endpoint-directory)
- Medicaid state endpoint file
- Payer endpoints - Medicare has a patient access API similar to the EHR API; health plans are required to provide these APIs similar to EHRs.

The minimum required information that needs to be included in endpoint lists is the FHIR endpoint base URL. Most lists also include an organization name for each endpoint, and Lantern will also parse zip code information from endpoint lists, if available.

The FHIR Capability Statements retrieved from these endpoints have the capacity to list software names and versions. However, inclusion of this data is inconsistent and does not clearly map to CHPL. If the list is from CHPL, the one or more software products associated with it are mapped to the endpoints from the list. Furthermore, the FHIR Capability Statements do not have the capacity to link the FHIR endpoint to an organization, so Lantern relies on the organization names and other organization data reported by the FHIR endpoint list data sources to link a FHIR endpoint with an organization. Details regarding the methods used to link endpoints to organizations are included in Section 6.

2 Developer Data

Developer data is parsed from the CHPL "**/developers**" API. Entries represent developers of certified health IT software products. The table below includes the list of fields that Lantern uses; additional fields can be found in CHPL's documentation.

Table 1. Fields Parsed from the CHPL Developers List

Field Name	Field Contents
id	Unique ID used within CHPL to identify developer
developerCode	Additional developer identification number
name	Name of the developer
website	URL of developer's website
lastModifiedDate	Date which the developer's entry was last modified
status	Indicates the active status of the developer
addressId	Unique ID of the address entry within CHPL
line1	Developer address line 1
line2	Developer address line 2
city	Developer address city
state	Developer address state
zipcode	Developer address zip code

country	Developer address country

3 Software Product Data

Software product data is parsed from the CHPL "**/search/v3"** API. Software products returned at this route represent certified health IT products that have been registered in the CHPL. The table below includes the list of fields that Lantern uses; additional fields can be found in CHPL's documentation.

Table 2. Fields Parsed from the CHPL Products List

Field Name	Field Contents
id	The CHPL ID of the developer who makes this software product
edition	The certification edition of this software product
product	Name of the software product
version	Version of the software product
chplProductNumber	Unique string used by CHPL to identify this software product
certificationStatus	Indicates if the software product is currently active
criteriaMet	List of CHPL criteria which this software product meets
certificationDate	Date that the software product was certified
practiceType	A practice type (either Ambulatory or Inpatient)
developer	The developer of the software product
apiDocumentation	Information about the documentation for the product

4 Data Validations

The Lantern system runs validations on the endpoints and stores the results in the **validations** database table. Lantern will run the set of base validations against all endpoints and will run FHIR version-specific validations depending on the version of FHIR advertised in the Capability Statement.

Table 3. Validation Result Table Format

Field Name	Field Contentsback
validation_result_id	Database id referenced by the endpoint in the fhir_endpoints_info table
valid	Indicates whether the actual value matched the expected value
actual	The actual value as reported by the endpoint
comment	Narrative explaining the validation
expected	Value(s) that will result in a passed validation
rule_name	Name of the validation
implementation_guide	Reference to an implementation guide (if any) relevant to the validation
reference	Link to relevant rule or standard that defines the expected value of the validation

Table 4. Base Validations

Validation Name	Validation Description
capStatExist	Asserts that a Capability Statement was returned by the endpoint
kindRule	Asserts that the Capability Statement's **kind** field has the value "instance"

describeEndpointRule	Asserts that Capability Statement includes a value for either the **description**, **software**, or **implementation** fields
documentValidRule	Asserts that if elements exist in the **document** field of the Capability Statement, that the documents listed are unique when keyed by the **document.profile** and **document.mode** fields
endpointFunctionRule	Asserts that the Capability Statement includes at least one **rest**, **messaging**, or **document** element
messagingEndptRule	Asserts that if the Capability Statement's **kind** field has the value "instance", then the **messaging** field should not be available
uniqueResourcesRule	Asserts that the list of resources advertised in the Capability Statement's **rest** field does not contain duplicate resources

Table 5. FHIR R4 Validations

Validation Name	Validation Description
patResourceExists	Asserts that the Capability Statement advertises support of the **Patient** resource
tlsVersion	Asserts that TLS version 1.2 or higher is used during transmission
otherResourceExists	Asserts that the Capability Statement advertises support for a resource in addition to the **Patient** resource
smartResponse	Asserts that the SMART Response resource is returned when querying the **/.well-known/smart-configuration** endpoint
instanceRule	Asserts that if the CapabilityStatement's **kind** field has the value "instance" then the **instance** field should be available
versionsResponseRule	Asserts that the default FHIR version as specified by the **$versions** operation should be returned from the server when no version is specified
searchParamsRule	Asserts that the names of search parameters within a resource are unique to said resource

5 Linking Mechanisms

5.1 Linking Endpoints to Developers

Most of the endpoints in Lantern are from endpoint lists in CHPL, so the developer is already associated with an endpoint list. Mapping in this case is simple since it is pulled from the CHPL entry and saved to any endpoint in the developer's list.

However, there are still lists in Lantern that are not from CHPL where the developer is not included. In this case, Lantern links FHIR endpoints to developers using developer names reported both in the **publisher** field of the Capability Statement and the CHPL **developers** list.

When a capability statement is received, the following matching steps are performed:

1. Normalize both the reported publisher from the Capability Statement and all of the CHPL developer names by converting all names to lowercase and removing any of the following words:

   "inc.","inc","llc","corp.","corp","corporation","lmt","lmt.","limited","corporation."

   Finish the normalization process by removing any trailing punctuation.

2. Iterate over the entire list of normalized developer names from the CHPL developers list. If the normalized developer name is a substring of the publisher's name or vice versa, then the developer is considered to be a match.

6 Query Intervals

Lantern queries its list of known FHIR endpoints once every 24 hours. Setting the query interval to once every 24 hours means that over time Lantern will have queried each endpoint at exactly same hour of the day. During each query Lantern records data from each endpoints' Capability Statement in addition to the HTTP response code and response time associated with the request made to the endpoint.

7 Endpoint Info History Pruning

The history pruning algorithm runs in parallel with the **Capability Querier** service, which queries endpoints and updates both the **fhir_endpoint_info** database table and subsequently the **fhir_endpoint_info_history** database table. The pruning algorithm first retrieves all distinct FHIR endpoint URLs from the **fhir_endpoint_info_history** table and processes each URL separately. For each URL, it examines entries that have **entered_at** dates older than the time determined by subtracting the environment variable LANTERN_PRUNING_THRESHOLD from the current time, and also have **entered_at** dates that are newer than a calculated lower bound, which is typically the current time minus the pruning threshold minus 7200 minutes (5 days). This time window approach ensures that the algorithm does not repeat pruning checks on the same entries after every query interval, but that it also does not miss any entries that have not yet been pruned.

The pruning algorithm also leverages a metadata tracking system that records information about each pruning operation in the **info_history_pruning_metadata** table. This allows the algorithm to resume from where a previous operation left off if it was interrupted or failed, ensuring complete processing of all eligible entries. The metadata system also tracks statistics such as the number of rows processed and pruned during each operation.

With the new implementation of history triggers, the system now only creates history entries when actual data changes occur in the **fhir_endpoints_info** table. For UPDATE operations, the trigger compares all significant fields between the old and new versions of a record using the `IS DISTINCT FROM` operator and only creates a history entry when differences are detected. This optimization significantly reduces the need for pruning by preventing duplicate entries at the source.

The pruning algorithm will remove any consecutive duplicate entries that may still exist in the **fhir_endpoint_info_history** table. A **fhir_endpoint_info_history** entry is considered a duplicate if there is an older consecutive entry that has the same stored information for the endpoint's TLS version, MIME types, and SMART response, and if the newer entry's stored Capability Statement only differs by fields included in a list of ignored fields, such as the **date** field. If a **fhir_endpoint_info_history** entry is found to be a duplicate of an older consecutive entry, it is deleted from the table, and this continues until only the oldest of the consecutive duplicated entries remains. Before deleting any validation entries, the algorithm checks if the validation ID exists in the current **fhir_endpoints_info** table to avoid removing data that might still be in use.

This combined approach of history triggers and regular pruning provides an optimal balance between data preservation and storage efficiency. The system maintains a comprehensive record of meaningful changes to endpoints while eliminating redundant data, allowing Lantern to effectively track how each endpoint has changed over long periods of time.

Appendix A Abbreviations and Acronyms

The list of abbreviations/acronyms includes all abbreviations, initialisms, and acronyms listed in the document.

Table 9. Appendix Terms and Definitions

Term	Definition
API	Application Programming Interface
CHPL	Certified Health IT Product List
CMS	Centers for Medicare and Medicaid Services
CSV	Comma-Separated Values
FHIR	Fast Healthcare Interoperability Resources
HTTP	HyperText Transfer Protocol
ASTP	Assistant Secretary for Technology Policy/Office of the National Coordinator for Health Information Technology (hereafter ASTP)
URL	Uniform Resource Locator