FHIR Architectural Patterns

Steve Munini
Sep 14, 2017 · 15 min read

A common misconception about HL7® FHIR® is that the standard is just an increasingly popular new healthcare data format. While it is true that the healthcare industry is rapidly adopting FHIR, viewing FHIR solely from the perspective that it is just a newer healthcare data format limits the impact FHIR can bring to your healthcare solution design. There are many new, compelling and powerful features that FHIR has to offer a software architect and developing an understanding is critical to building robust solutions and products with FHIR.

FHIR, fundamentally, is a “platform specification” that provides solutions to many common challenges encountered in describing, managing, sharing, and using clinical data in numerous complex contexts that are found in healthcare. The developers of HL7 FHIR have made some deliberate design decisions when creating FHIR that are huge steps forward in working with healthcare data. I have described many of them here in our first blog article.

This article looks at the many ways in which FHIR can be used when designing a solution, or an “Architecture Pattern.” After reading this article, you will find that FHIR not only supports common interoperability challenges but, because of its design as a platform specification and many of FHIR’s interesting features, FHIR opens the door to new, modern, flexible and adaptive sets of healthcare product solution designs. These newer types of FHIR-based architecture patterns are described and I also describe some of the design principles of our own FHIR server.

FHIR’s Assumption-Less Design Approach*

FHIR makes no assumptions about the design of systems in which it will be used. We can use FHIR to build:

- A lightweight client such as a mobile phone app, or a heavyweight client such as a robust analytics healthcare data solution
- A central server architecture or a peer-to-peer sharing solution
- A push or a pull-based design
- A query approach or publish/subscribe
- A loosely or a tightly coupled system
- Systems that require change history tracking, or those that do not

FHIR does not impose a specific use model on the solution you are aiming to design. Also, picking one approach does not exclude other approaches. It is also possible to use multiple FHIR architectures in a single solution design. For example, one solution we have worked on uses FHIR to retrieve clinical data from an EMR (the Interoperability Interface pattern). The solution also needs to perform analysis on genomic data, so we are exploring the use of a separate, genomic FHIR database endpoint (the Rapid FHIR-Based Endpoint pattern) which will store clinical genomic sequence information.

Two Approaches to FHIR

There are two fundamental ways to approach the use of FHIR in your solution architecture. The most common use of FHIR is to add a FHIR API to an existing solution. I cover this scenario first, and then we will look at some new approaches to designing solutions using FHIR.

FHIR-Enabling an Existing Solution

In this approach, software architects start with an existing solution and work towards making their product or solution capable of offering a FHIR-compliant API. These solutions come in the form of:

- EMR Vendors
- Healthcare Data Interchange Solution Providers
- Claims Systems
- Pharmacy Systems
- Medical Terminology Systems
- Other existing solution vendors

The path to FHIR-enable an existing solution involves learning the FHIR specification, understanding the existing solution's desired use of FHIR, selecting a FHIR paradigm, mapping the existing solution's data model to FHIR Resources, specifying the solution's FHIR capabilities in a Capability Statement, and then, of course, testing the solution.

The task at hand is largely a mapping effort as result, and most of the time, these solution providers will never, by design, fully support the entire depth and breadth of the FHIR specification. That's okay — FHIR was designed to be used incrementally. Some solutions, for example, medical terminology products, only ever need to offer a subset of the complete FHIR specification. Medical terminology servers are focused on providing the ValueSet resource, as well as their supporting operations, and may not need to support many of the other FHIR resource sets.

Another example of partial FHIR specification support are EMR vendors themselves. At this current stage in EMR adoption of FHIR, many EMR vendors do not yet support write capabilities on their FHIR interfaces — they only support reads. This may change with time as the industry is rapidly moving to FHIR-based systems, and EMR vendors are being encouraged by their customers to support FHIR writes. Supporting write capabilities by an EMR vendor can be a complex undertaking, especially for those vendors who have application logic and other subsystem capabilities that are triggered to a write event.

FHIR As a Next-Generation Health Data Platform

Another FHIR architecture option exists in the form of using FHIR as the central design element of a new type of healthcare data technology. FHIR is fundamentally a platform specification, and as such, there are many powerful features that makeup FHIR as a platform solution including:

- **Structure Definitions** — Describe FHIR constructs themselves — a meta-language for FHIR. Structure Definitions are used to describe the content defined in the FHIR specification — Resources, data types, and underlying infrastructural types.
- **Search Parameters** — Describe the types of search capabilities that are supported by a FHIR server.
- **Capability Statement** — A way for a FHIR server to "advertise" its set of capabilities to other software systems. A client may simply examine a FHIR server's Capability Statement to determine, for example, if a particular FHIR Resource is supported and if it can expect to read and write those resources to the server.
- **Healthcare Domain Model** — A set of well-defined and well-thought-through resources that describe the domain of healthcare. In the FHIR model, you will find concepts such as a patient, medication, allergy, and many other constructs defined robustly.
- **Extensions** — A built-in way to extend, or add to a FHIR resource, or add to specific elements within a FHIR resource.
- **Profiles** — Because FHIR is a platform specification and is quite flexible, often the need arises to specify which parts of the FHIR specification are to be used. Using a Profile, a solution designer can specify which resource elements are used and which are not, which terminologies are used in certain elements, and also how the API may be called.

These are just some of the features of FHIR and is not intended to be a comprehensive list, but as you can see, with such a well-defined set of platform constructs, it becomes possible to build an entirely new healthcare data product based upon the FHIR specification itself.

This is the design path that we have followed with the Helios FHIR Server. We have taken the FHIR specification, and have used it to create an entirely new clinical data product on top of the highly scalable, distributed, and extremely fast open-source Apache Cassandra™ database.

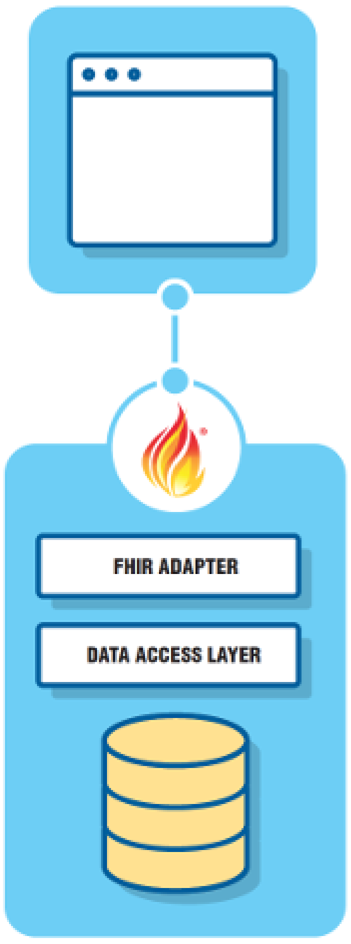
We have started with FHIR Structure Definitions and have used them to generate a Java object model, Cassandra table structures, and user-defined data types, REST API controllers and data access methods. Additionally, we use FHIR-defined Search Parameters to create the necessary indexes that support FHIR searches in a very fast and scalable manner that Cassandra provides. In a future blog post, we will have a deeper look at the internal design of the Helios FHIR Server.

8 FHIR Architectural Patterns*

Below, I have identified 8 FHIR architectural patterns. Several of these patterns are quite obvious and describe the types of common challenges that FHIR was designed to solve. Other patterns are a bit more advanced and will make you think about the kinds of new products that may arise in the marketplace now that FHIR is here and is being adopted in the industry.

The first four of these patterns fall under the FHIR-Enabling an Existing Solution approach and the last four fall under the FHIR As a Next Generation Health Data Platform approach.

Interoperability Interface



Intent: Supply an interoperable API on top of an existing solution that can be easily understood and consumed by clients.

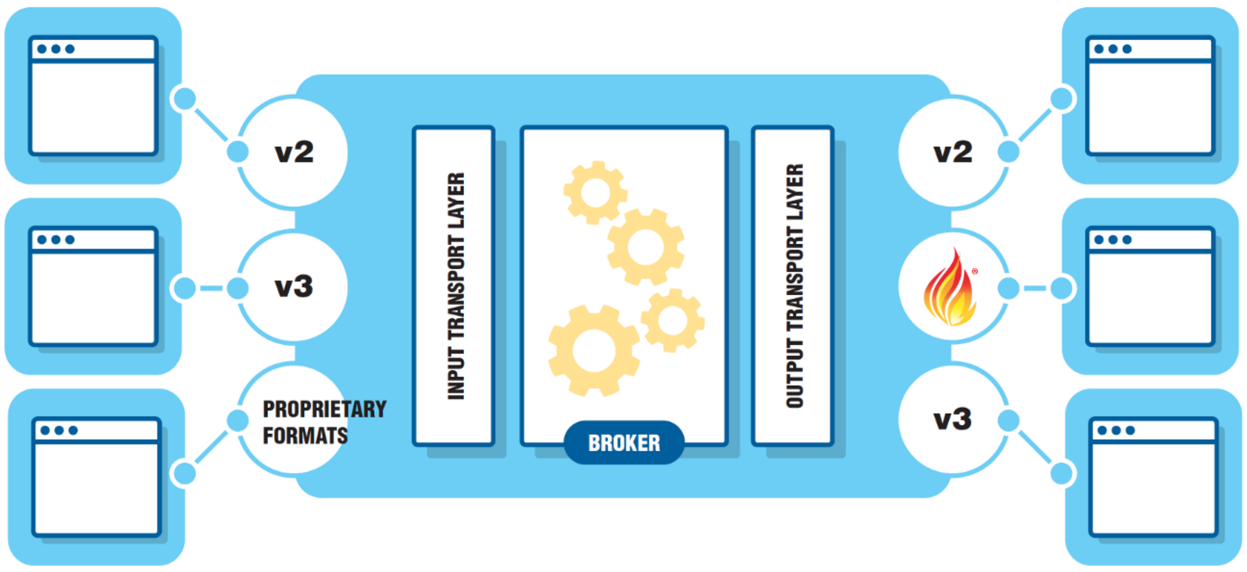
The interoperability interface architecture pattern is the most common pattern we see. Note that I didn't call this pattern an EMR interoperability interface, because this pattern is not limited to EMR vendors alone. Many other types of healthcare solutions and vendors adopt this pattern too. For example, terminology vendors use this pattern and provide a FHIR-compliant API that can be used by many other systems in a healthcare computing environment.

Participants:

- Clinical Database: An EMR's main database (for example, Epic uses Cache), a relational database such as Oracle, Microsoft's SQL Server, MySQL, Postgress, or even a nonrelational database such as Apache Cassandra.
- Data Access Layer: A layer of software that knows how to access and understand the proprietary clinical database structure and present it to the next layer of software.
- FHIR Adapter: Software that adapts, or otherwise translates data from a proprietary database to FHIR resources.
- FHIR API: HL7's FHIR API supplying services most commonly in a RESTful API paradigm, or other supported paradigms such as Messaging, Document, or Services paradigms.

A note about the FHIR Architectural Patterns and FHIR Paradigms. I view these two as topics as orthogonal. In other words, one can use a Messaging paradigm in an Interoperability Interface architecture in one solution, and a RESTful paradigm in a FHIR Broker Adapter architecture in the next. Another implementation may flip these paradigm and architecture choices.

FHIR Broker Adapter



> *Intent: Convert industry-standard healthcare data formats and proprietary formats to and from FHIR "on-the-fly."*

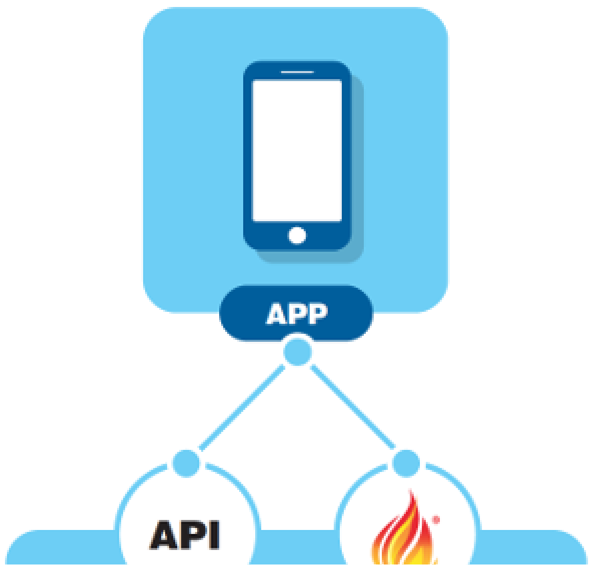
Healthcare computing environments are complex. Data resides in many different systems, even when there is a single, unified EMR installed. Also, data often needs to go to organizations and systems outside of the four walls of a hospital or healthcare organization in a variety of different formats, using myriad transport methods.

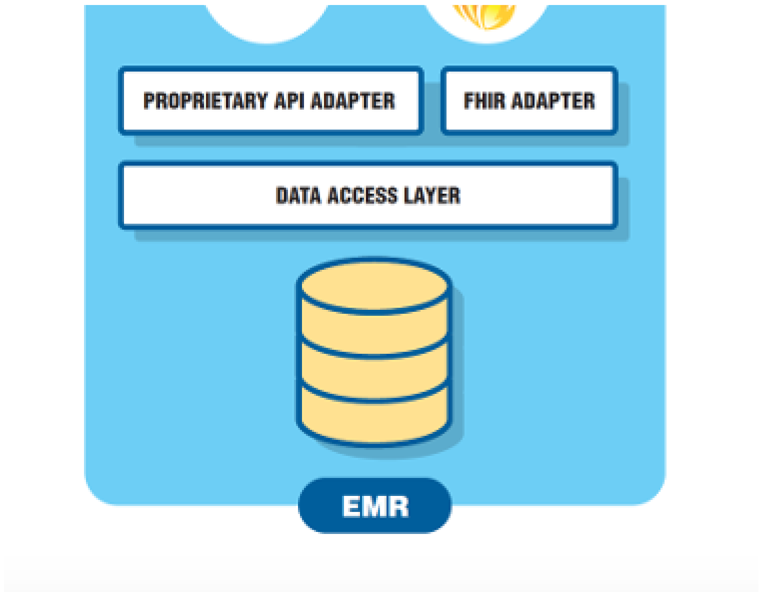
Healthcare organizations need to send data to and from many systems such as:

- Bi-directional data exchange with other provider groups and payers
- Participate in a health information exchange (HIE) networks
- Integrate with lab orders and results
- Send data to state immunization registries
- Send data to syndromic surveillance public health agencies

Several interoperability vendors in the market do this work exclusively — it is a specialized area of expertise that takes excellent technology, talented staff, and exceptional operational support to do it well. These vendors sometimes do not persistently store the messages, or if they do, they do not store them forever — these systems are transfer systems — and are not necessarily designed to be permanent long-term storage. FHIR is simply another format with which to transform.

FHIR/Proprietary API Mixed-Use





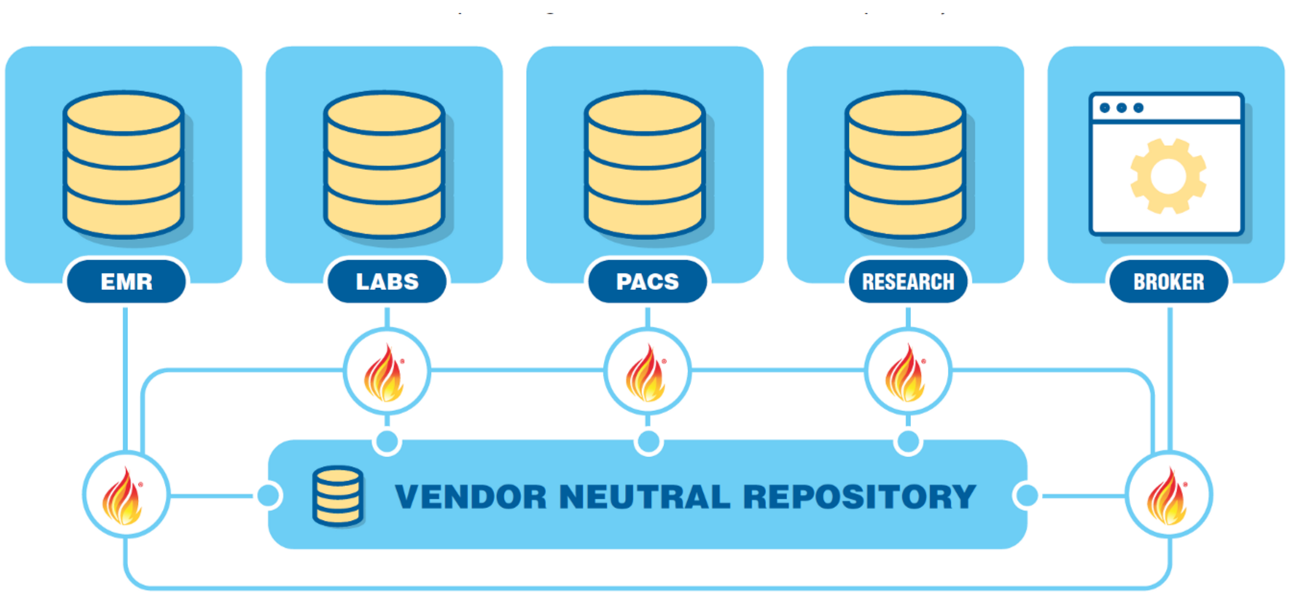
Intent: When building a production application that may need to interoperate with several EMRs, use FHIR to achieve code reuse and EMR portability, but also use proprietary APIs to satisfy business requirements.

This architecture approach goes against the grain of how a software architect wishes to ideally design a system. In an ideal design world, a solution would have a complete and comprehensive functional separation layer, simplifying the design and thereby providing an elegant solution.

Reality often steps in when designing real-world systems. There are two very real reasons why your solution may need to use this architectural pattern.

1. The FHIR server with which you intend to integrate doesn't support the FHIR capabilities you need. A common case here is FHIR write support. Several large EMRs do not support FHIR writes, yet your application solution may have mandatory requirements to store all data in the EMR for compliance purposes. Should a hospital, for example, receive a request for all of a patient's data, the release of the information department is only going to look for data in one location — the EMR. It is complex, cumbersome, and unrealistic to have that department probe numerous systems for data about a patient — especially when they might not even know about the existence of those other systems.
2. The FHIR specification may not describe your healthcare domain concept or use case. If you find this to be the case, consider participating in HL7 to advance the FHIR specification itself! You will find an interested community of other healthcare technologists that want to help advance the state of interoperability along with you.

FHIR API Encapsulating a Vendor-Neutral Clinical Repository

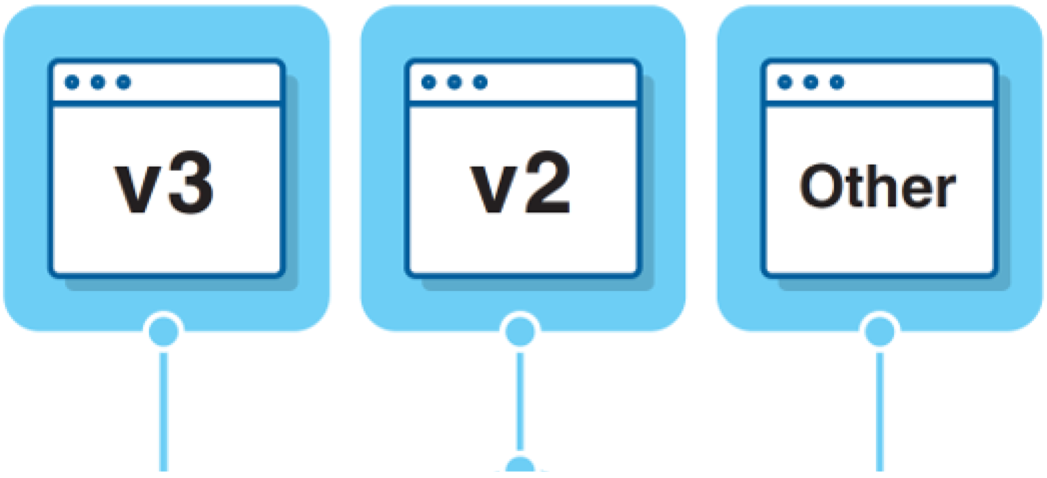


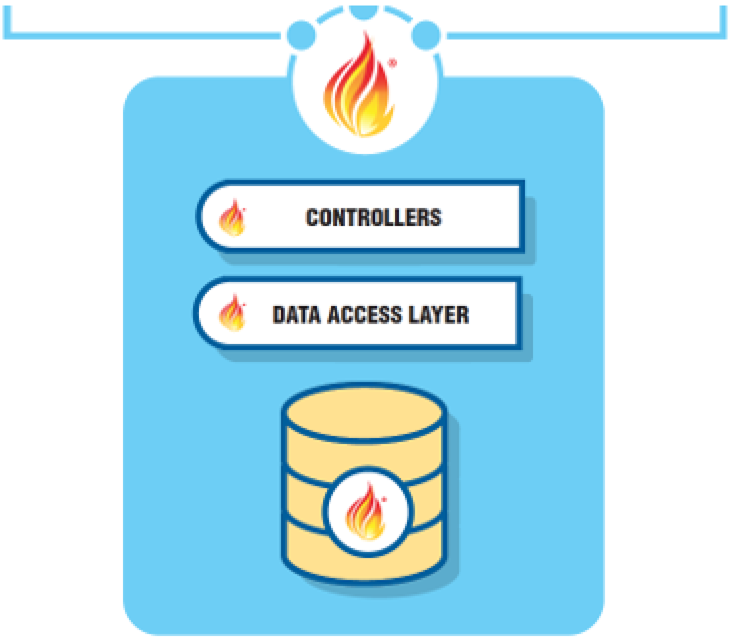
> *Intent: Use FHIR as a consistent API to a vendor-neutral repository, increasing substitutability of systems and increasing consistency of data access.*

A common requirement in healthcare organizations is the need to collect many types of data into a single location for numerous reasons, for data warehousing, or analytic projects for example. Often the EMR itself is not a feasible or logical product choice to perform this type of work.

FHIR has a deep domain model and excellent extension capabilities and profile support to satisfy these requirements.

FHIR-Based Clinical Data Repository





> *Intent: Instead of mapping a solution to fit FHIR, start with FHIR as a platform design specification, and create a FHIR-native solution from it directly.*

This architectural pattern takes the opposite approach than the previous four patterns. In this architectural pattern, FHIR isn't a specification to which an existing solution would need to be mapped or adapted. Instead, this architectural pattern begins with FHIR as the key design centerpiece of an entirely new product.

I wrote earlier about some of the very powerful FHIR constructs that make this architectural pattern possible, such as Structure Definitions, Extensions, Search Parameter definitions and Profile support, so I won't repeat them here, however it is important to understand these constructs to understand the design approach to this pattern. These constructs and the right set of software tooling that can convert these FHIR constructs to working code make it possible to not only generate a FHIR-compliant system from those artifacts but also create a much more flexible healthcare data solution generally.

This is the model of the Helios FHIR Server. We have used FHIR as the key design construct for an entirely new product. Let's look at just one element of what the FHIR specification defines, and then see how that translates to working code.

Below is a snippet of the FHIR specification that describes a SearchParameter — a way to locate resources by certain values. In this example, the "individual-email" search parameter enables us to find any of the 5 resources at the bottom of the XML example, Person, Patient, etc. by their email address.

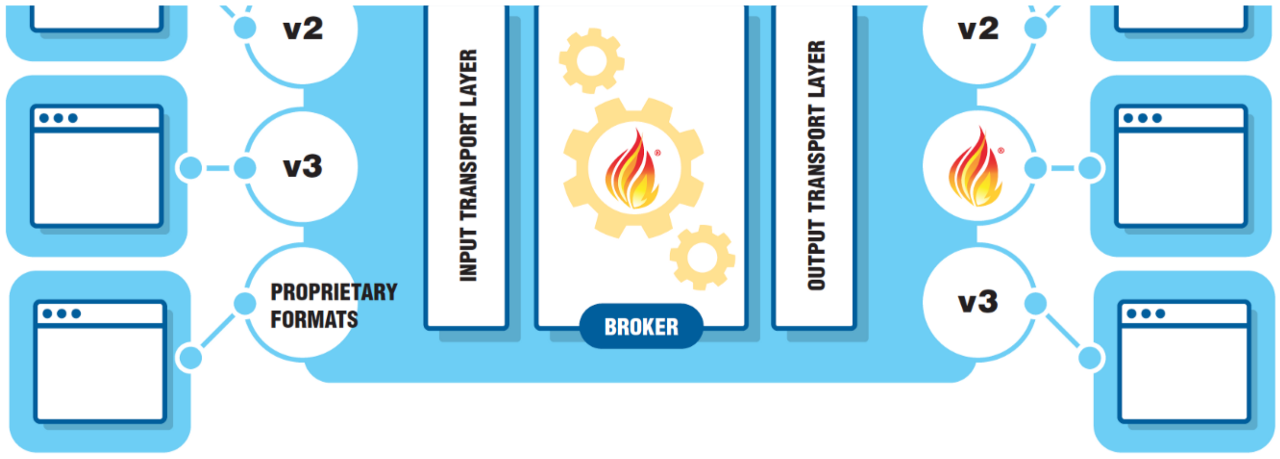
```
<SearchParameter>
  <id value="individual-email"/>
  <url value="http://hl7.org/fhir/SearchParameter/individual-email"/>
  <name value="email"/>
  <status value="draft"/>
  <experimental value="false"/>
  <date value="2017-04-19T07:44:43+10:00"/>
  <publisher value="Health Level Seven International (Patient Administration)"/>
  <contact>
    <telecom>
      <system value="url"/>
      <value value="http://hl7.org/fhir"/>
    </telecom>
  </contact>
  <contact>
    <telecom>
      <system value="url"/>
      <value value="http://www.hl7.org/Special/committees/pafm/index.cfm"/>
    </telecom>
  </contact>
  <code value="email"/>
  <base value="PractitionerRole"/>
  <base value="RelatedPerson"/>
  <base value="Practitioner"/>
  <base value="Person"/>
  <base value="Patient"/>
```

Using the FHIR SearchParameter, along with other elements of the FHIR specification, we can generate not only the necessary database table structures, and domain object model, but also a Cassandra query such as the one below that will locate all Patients with an email address matching the "?" parameter.

```
"SELECT * from Patient_by_email WHERE email = ?"
```

Products that use this architectural pattern provides incredible flexibility. It certainly future proofs any solution that may wish to integrate with FHIR as the standard evolves, but it also provides a framework for extending and enhancing solutions. Should your solution need to add another search parameter that, perhaps the FHIR standard does not specify — simply add another SearchParameter to your environment, and the necessary indexes and supporting access code is generated.

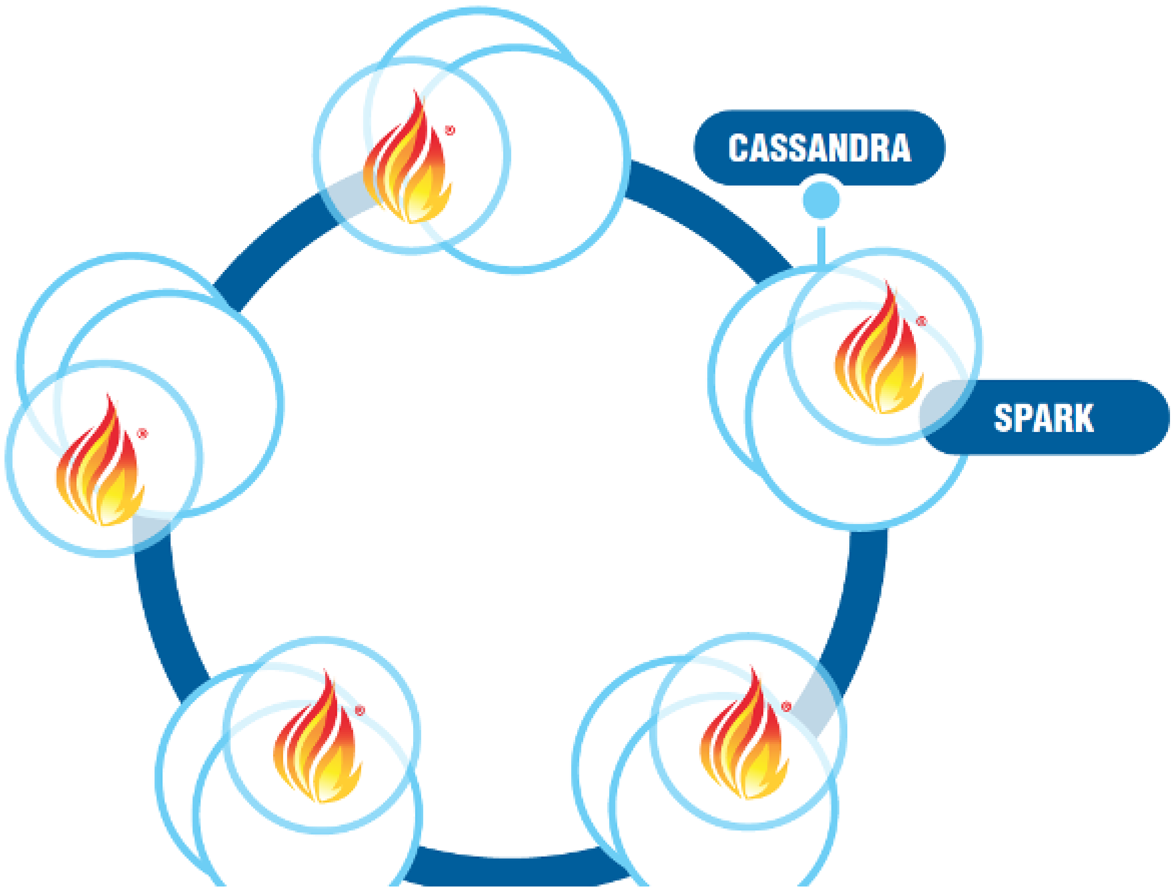
FHIR-Based Integration Hub



> *Intent: Use FHIR's domain object model as the native interchange format from which to map a variety of other formats both to and from FHIR.*

Using A native FHIR object model as the basis for a new type of healthcare broker solution has a key advantage. By using FHIR internally as the underlying domain model, this shifts the responsibility of a broker software vendor from maintaining their own internal proprietary format and mapping to another mapping — simplifying the design of these solutions greatly.

FHIR-Based Analytics Solution



> *Intent: Instead of transferring data to a separate analytics solution, keep the data in place, and use FHIR as the underlying domain model upon which to perform analysis.*

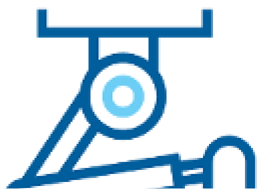
Online analytical processing systems are those that are configured to use a multidimensional internal data model, allowing for complex analytical and ad hoc queries. These systems are very powerful and have been designed to produce very fast answers to queries on large data sets using a cube-oriented data model architecture. This involves populating cubes from a separate data source which, if not done properly, can introduce data lag (additional time required to get the data from an operational data store to the analytical data store) and synchronization issues (is the correct data available for analytics processing?) in a solution design.

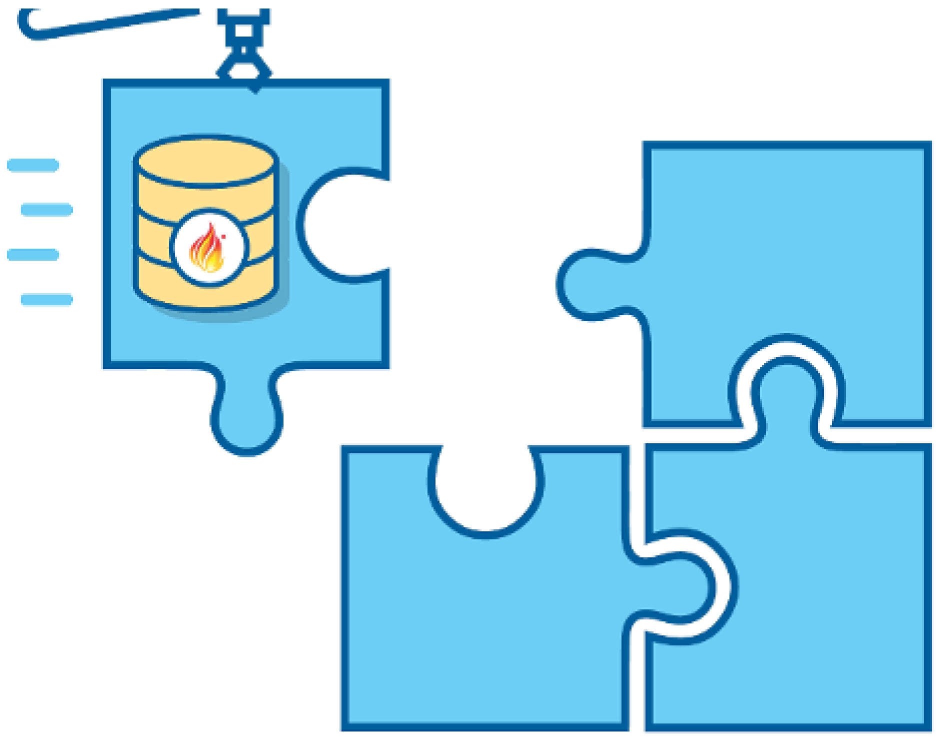
Using FHIR as the underlying object model and generating a database (or a cluster of nodes in the case of Apace Cassandra) optimized for analysis can be a high-performance solution to some of the challenges with running a separate OLAP system.

Increasingly, the industry is considering a different approach to analytic processing. One approach is to place processing nodes co-resident with the data storage nodes in a distributed fashion. This is exactly the approach taken with Apache Spark (processing) and Apache Cassandra (storage). The fundamental idea is very straightforward: Spark and Cassandra clusters are deployed to the same set of machines where Cassandra is responsible for storing the data and a Spark worker node, running on the same machine is responsible for data processing.

Spark is a system that is designed to work with large amounts of data in a batch-oriented fashion. When a task arrives, Spark loads data into memory directly, and since the Spark node is co-resident on the same node as Cassandra, there isn't network traffic impact that often slows such analytics systems considerably.

Rapid FHIR-Based Endpoint





> *Intent: Provide a FHIR-compliant healthcare data storage solution rapidly.*

FHIR is an excellent integration technology. Sometimes, however, it may be faster, and easier to not use FHIR as an integration technology, but instead, use it as a standalone database. There are many real-world scenarios where it may be faster, easier, and less costly to simply stand-up a FHIR database and begin working with it immediately.

- Prototypes — FHIR is new, and there are many ongoing efforts involving early prototyping with FHIR to better understand it's capabilities and features. Installing a standalone FHIR-compliant database is a great way to begin prototyping.
- Startups — Healthcare startups often have a large design challenge initially in terms of simply describing the data with which they want to work. For example, if a healthcare startup company is building a solution involving medications, using FHIR as their medication data model is a very easy way to make rapid progress.
- No means to store data otherwise —One of FHIR's key strengths is its expressive set of foundational resources that enable us to describe healthcare concepts easily and rapidly. Often, when building healthcare products, it is much easier to borrow the tooling of a rapid FHIR-based endpoint than it is to re-create it from other means. For example, if your solution finds that it needs to begin to work with genomic data, and you don't have a genomic database handy, and don't want to build one —

a rapid FHIR-based endpoint that can both read and write FHIR sequence resources would be a great place to start.

- Easier to store it here, rather than build an adapter there — Sometimes it is just simpler to store data in a temporary location, in the format that you desire, rather than investing the additional development time that may go into building an adapter. Often, it is a question of people too — the development team that may be required to accomplish an adapter development effort may not be available in the timeframe you need.

Summary

With this article, I have aimed to show you some of the many ways that you can use FHIR in your solution designs, or FHIR Architectural Patterns. There are two fundamental approaches to using FHIR. The first is adapting your solution to fit FHIR by providing FHIR as a new interface on top of an existing solution. This is the most common approach when thinking about using FHIR. I have also outlined an entirely different approach to using FHIR, which is to use the FHIR specification, and all of its expressiveness to create FHIR-native solutions. FHIR-native solutions offer a great amount of flexibility, future-proofing, speed of development and integration, and are a promising new approach to working with healthcare data. Our own Helios FHIR Server follows this design approach.

An Offer For You

If you have found this blog post to be of interest to you, please do share it on your favorite social media platform.

Additionally, we have created a group on LinkedIn of the industry's leading Health IT developers and architects who are building high-performance analytic healthcare applications using FHIR. If topics like the ones I discussed in this blog article are of interest to you, please consider joining our group.

LinkedIn Group

Our LinkedIn group for Health IT professionals brings together over 2,300 leading developers and architects who build high-performance analytical healthcare applications using FHIR.

Email

First Name | Last Name

Sign up

I agree to leave Blog.heliossoftware.com and submit this

* Credits

Thanks to Justin Mitchell.

Healthcare Fhir