A TARK FIELD GUIDE

Choosing Your EHR Integration Standard

FIELD GUIDE

FHIR R4, HL7 v2, and C-CDA: how to pick the right one for each interface, and the mistakes that quietly cost teams months.

START HERE

The choice that quietly decides your timeline

Ask three engineers which standard to use and you will get three confident answers, usually the one they used last. That habit is expensive.

The standard you pick for each interface shapes how fast you ship, how much your team re-keys, whether the connection survives the next EHR upgrade, and whether write-back works at all. This is how we decide, interface by interface, on real projects. It is opinionated on purpose, because "it depends" is not a plan.

Three standards do most of the work in US healthcare. None is the answer on its own. The skill is knowing which one earns its place where.

HL7 v2

THE WORKHORSE

If healthcare data has moved in the last three decades, HL7 v2 probably carried it. It is a real-time messaging standard, pipe-delimited and terse, and it still runs the busiest lanes in the hospital: admissions, discharges and transfers (ADT), orders, and results. It travels through interface engines that most health systems already operate, which is exactly why it stays the fastest path for those flows.

Its strengths are ubiquity, real-time delivery, and mature tooling. Its weakness is conformance. HL7 v2 is a standard the way pizza is a food: every site has its own recipe. Two "standard" ADT feeds from two hospitals will differ in segments, fields, and meaning, and the spec permits it.

FIELD NOTE

We treat every HL7 feed as bespoke until proven otherwise. The conformance profile a vendor hands you describes intent, not what is actually on the wire. We validate against real messages before we design, because the week you save skipping that step is the month you lose re-mapping later.

FHIR R4 (US Core)

THE MODERN API

FHIR is what integration would look like if it were designed today: RESTful, resource-based, JSON, and friendly to web and mobile developers. With SMART on FHIR and OAuth 2.0, your app can launch inside the EHR in the context of the current patient and clinician. For modern reads such as problems, medications, allergies, and results, and for anything patient-facing, it is the right tool.

The trap is assuming FHIR write is as available as FHIR read. Across the major vendors, read coverage is broad while write coverage is uneven and version-dependent. A workflow that reads beautifully in the sandbox can hit a wall the moment it needs to write back.

> **FIELD NOTE**
>
> Before we promise write-back, we confirm it against the specific vendor, version, and environment, not the FHIR spec in general. The spec is a ceiling, not a guarantee. "Supported resource, unsupported field" is where confident timelines go to die.

C-CDA

THE DOCUMENT STANDARD

When the unit of exchange is a whole clinical picture rather than a single data point, C-CDA is the standard: structured XML documents such as the Continuity of Care Document, used for transitions of care and regulatory exchange. One artifact carries a rich snapshot of the patient.

It is comprehensive and heavy. The XML is dense, authoring varies widely between systems, and technically valid documents can still be painful to turn into usable data. Teams routinely underestimate the extraction work.

> **FIELD NOTE**
>
> We parse defensively. Real C-CDAs from the wild rarely match the tidy examples in the documentation, so we build for the messy ones from day one and treat the clean sample as the exception, not the rule.

DECIDE PER INTERFACE

When to use which

Most real integrations use more than one standard. The goal is not loyalty to a standard, it is the right tool for each interface.

- **Orders, results, ADT, real-time messaging.** HL7 v2 through the interface engine the site already runs.
- **Modern clinical reads, app launch inside the EHR, patient-facing and mobile.** FHIR R4 (US Core) with SMART on FHIR.
- **Whole-record summaries, transitions of care, regulatory document exchange.** C-CDA.
- **Write-back.** Verify per vendor first. Often FHIR where it is genuinely supported, HL7 where it is not.

Side by side

STANDARD	BEST FOR	SHAPE	WATCH OUT FOR
HL7 v2	Orders, results, ADT, real-time	Pipe-delimited messages via interface engine	Site-by-site variation; loose conformance
FHIR R4 (US Core)	Modern reads, app launch, patient-facing	RESTful JSON resources, OAuth and SMART	Uneven write support; version drift
C-CDA	Clinical summaries, transitions of care	Structured XML documents	Heavy parsing; inconsistent authoring

FIELD NOTE

The strongest integrations are rarely single-standard: FHIR for reads, HL7 for orders and results, C-CDA for history. One standard for everything usually means the interfaces have not been mapped yet.

HARD-WON

The mistakes that cost months

None of these are exotic. They are the ordinary assumptions that look harmless in a kickoff and resurface as a blown quarter.

01 **Treating FHIR read and write as equal.**

Read coverage is broad; write coverage is not. Confirm write, per vendor and version, before you scope around it.

02 **Assuming HL7 feeds are interchangeable.**

Every site is its own dialect. One integration is not automatically reusable at the next hospital.

03 **Underestimating patient matching.**

Wrong-patient data is the failure mode nobody budgets for. Identifiers and matching logic deserve design time, not an afterthought.

04 **Building to the spec, not the implementation.**

The published profile and the live endpoint are different animals. Design against the one that actually answers.

05 **Skipping monitoring and retries.**

Interfaces fail quietly. Without logging, alerting, and retries, the first sign of trouble is a clinician's complaint.

06 **Designing for the happy path.**

Real feeds carry malformed messages, missing fields, and duplicates. If the design assumes clean data, production will correct that assumption for you.

HOW WE DO IT

How we choose the standard on a real project

Here is the sequence we run, and why each step exists. None of it is glamorous. It is the difference between an integration that holds for years and one that needs a rescue in month three.

01 **Map the data, both directions, per interface.**
What you read, what you write, and where it has to land.

02 **Check the vendor's actual API, version, and environment.**
Not the spec in the abstract. The endpoint that will answer your calls in production.

03 **Pick the standard per interface for reliability, not habit.**
Often more than one across a single product.

04 **Prove it in the sandbox with real messages.**
Before a timeline is committed, not after it slips.

05 **Build with retries, logging, and monitoring from sprint one.**
So the first sign of a problem is an alert, not a phone call.

Deciding on standards for a specific build?

That is the conversation we have best. Tell us what you are connecting and to which system, and we will map the right approach, interface by interface. Reach us at **hello@tarktech.com**.