# Comparison of V2, V3, CDA and FHIR

The following table compares and contrasts the v2, v3 messaging, CDA and FHIR specifications, highlighting the things each does well and what each does poorly. It may help in an evaluation of the respective specifications and highlights some of the benefits of FHIR. The color-scheme is informal. Green indicates “close to ideal state”. Orange indicates “far from ideal state”. Yellow indicates “somewhere in between”.

| Characteristic | V2 | v3 messaging | CDA | FHIR |
| --- | --- | --- | --- | --- |
| *Capability* |  |  |  |  |
| Breadth of coverage | Administration and diagnostic-focused | Pretty much any healthcare scenario  you can imagine | Limited to “common”, patient-specific clinical scenarios | Pretty much any healthcare scenario  you can imagine |
| International Scope | Significant US-specific content | International | Mostly international (some popular templates US-centric) | International |
| Communication Paradigms | Messaging only | Messaging, some Services | Documents | Messaging, Documents, Services, REST |
| Support for complex scenarios | Limited data types and structures. Limited complexity handled in extensions | Robust support for complexity | Support for complexity in “in-scope” areas | “uncommon” complexity handled via extensions |
| Human readable content | Sometimes – by profile in NTE segment | Not usually | Yes | Yes, when desired |
| Extensibility | Yes (Z-segments) | Sort of (foreign namespace or special attribute) - discouraged | Sort of (foreign namespace) - discouraged | Yes (extensions section) – commonplace |
| Unknown extensions understandable? | No (unless you have analyst phone #) | Usually (if RIM naming followed) | Usually (if RIM naming followed) | Yes – via URL of extension |
| Out of the Box interoperability | Rarely. Interface engines required | Almost never at Int’l level – significant realm constraint needed | Yes – for human-to-human, simple metadata; templates required for system to system | Yes for REST, documents and simple messages if no “must understand” extensions invoked |
| *Architecture* |  |  |  |  |
| Object-oriented | No | Yes | Yes | Yes |
| Discrete, re-usable, context-independent components | Yes (Segments) | Sort of (CMETs) | Sort of (Entries) | Yes (Resources) |
| Robust semantics | No | Yes (not always modelled well). Intrinsic to presentation | Sort of (quality depends on template, some semantics inexpressible) | Yes (for resources), where possible for extensions) |
| *Wire format* |  |  |  |  |
| Modern / well supported wire format | Character delimited, some XML | XML (theoretical others) | XML | XML (JSON option) |
| Wire format backward/forward compatibility | Yes | No | Partial (within releases, not always between releases) | Yes |
| Human readable wire syntax? | Low – count vertical bars, guess based on data values | Low – bloated XML, formal names non-intuitive | Low – bloated XML, formal names non-intuitive | High |
| Size | 2304 pages (v2.7) | >> 10k pages | 200 pages (+ ~400 of infrastructure) | ~1000 pages (estimate) |
| Learning curve | Moderate | Very high | High | Moderate |
| *Conformance* |  |  |  |  |
| Conformance profiling | As of 2.5, not commonly used | Conformance rules defined, generally handled via constrained constrained artifacts | Conformance rules defined. Templating only, typically captured as Word, though some formal capture, no formal declaration mechanism | Yes - profiles computer processable |
| Conformance declaration | Done using profiling mechanism | no formal declaration mechanism | no formal declaration mechanism | Yes – conformance statements computer processable |
| *Additional considerations* |  |  |  |  |
| Modeling expertise needed for design | None | High – key part of design | Moderate – needed for good template creation | Moderate – needed in parallel with resource and extension creation |
| Modeling expertise needed for implementation | None | Moderate – model is central in all documentation, non-fixed structural codes must be populated correctly | Moderate – model is central in all documentation, non-fixed structural codes must be populated correctly | None (though information available for those wanting it) |
| Standard creation tools | MS Word, validation / extraction via custom MS Access | Numerous custom applications, Windows-specific, custom generation tool | Same as v3 with manual editing of output of generation tool; additional custom tools for template authoring | Excel or Open Office, Enterprise Architect?, custom generation tool |
| Publishing approach | One spec, document form | Multiple interdependent specs, web-based | One main spec, few additional specs, web-based | One spec, document or web-based |
| Code generation | Some open source tools | Some open source tools | ??? | Part of published specification |

# What v3 issues does FHIR address?

The following table identifies a number of the issues that make implementing v3 specifications “hard”. For each, it identifies how much FHIR addresses the issue and provides descriptive comments about how it is addressed (or not)

| **Issue** | **Addressed?** | **Comment** |
| --- | --- | --- |
| Wire format hard to read | Mostly | Core content is easy to read. Extension content automatically convertible to be easy to read |
| Specifications hard to read/navigate | Mostly | Specification volume significantly reduced, focus is on “easy to read for implementers”, RIM modeling is suppressed |
| RIM experts hard to find | Mostly | RIM understanding no longer needed to read the spec or as critical path for resource or extension development. Still need RIM expert to do mappings, but can be more efficient and leverage fewer resources |
| Poor QA on v3 specifications | Partially | Smaller volume of elements (countable number of resources, ~80% reduction in data elements combined with stronger governance should allow better QA. As well, the requirement for instances and the expectation of committees to build (and thus validate) frequently should help too. Work Groups will be strongly discouraged from checking in content that fails validation.  Voluntary HL7 vetting process for extensions should help quality there too. |
| Non-interoperability across borders (project/geographic) | Partially | Wire format is the same everywhere – all countries, all domains. However, constraints can be different and some extensions may be “must understand” which can still interfere with interoperability. |
| Lack of implementer support | Partially | Specifications targeted to implementers. Code interfaces generated “out of the box” for common platforms. Specification tuned to address specific implementation issues on those common platforms. Implementation tests early in the process and Connectathon early in the process.  Will still need broad outreach to have implementers actually pay attention and engage early enough in the process. |
| Local variation of data elements | Partially | With FHIR, all extensions are sent in a consistent way that doesn’t break schemas. And you’re expected to accept unrecognized extensions. So long as an extension isn’t “must understand” you can create an app that populates all extensions everyone needs and send everywhere and it’ll work. |
| Local variation of constraints | No | If you make an element required, you’re not going to inter-operate with someone who has it as optional until they change their code. Same issue with minimum/maximum number of repetitions. |
| Local variation of terminology | No | If you send different codes than receiver requires, you’re not going to interoperate. |
| Terminologies are hard | No | FHIR doesn’t eliminate issues around complex terminology hierarchies, post coordination, etc. |
| Gaining consensus takes a long time | Tiny bit | In theory, a resource that’s at the 80% allows the consensus making to focus on extensions rather than on core, though there’ll still be a need to discuss constraints on core. And consensus is always hard. |
| Hard to make local changes quickly and push up | Partially | With FHIR, it becomes possible for a local project to throw together resources and extensions with little modeling expertise on a short timeline and still be reasonably aligned with a clear transition path to pan-Canadian equivalent that may take much longer to get resourced. Changes will be for RIM mappings (transparent to most implementers) and possible changes to some of the extensions. (manageable) |
| Content not shareable across paradigms | Yes | Content is represented the same way in documents, messages, services and REST. (Documents **may** supplement with additional organization/rendering structures.) |
| Creating generic templates (usable across multiple specifications) is hard | Mostly | With FHIR, there’s a single base model for each clinical concept – the resource. It has fixed element names. Profiles written against that resource can apply against all instances, regardless of jurisdiction, discipline, domain, etc. Still an issue for templates intended to apply across resources, but countable number of resources makes it manageable. |
| Proliferation of templates/profiles | Not really | FHIR makes it easy to create and roll out template (profile) registries. But proliferation is a governance issue, not a technical issue. |
| DCM consensus is hard | Tiny bit | FHIR - due to extensible, discrete and x-paradigm nature provides good foundation for DCMs. However, consensus process and issue of “boiling the ocean” (extremely large number of needed DCMs) |
| Interoperable discrete data capture impacts application design | No | Interoperability means capturing discrete data in standardized ways using standardized vocabulary. There’s no way for this to avoid impacting (and often significantly impacting) application design. |
| Interoperability affects workflow | No | If the workflow isn’t standardized, your applications won’t interoperate. FHIR supports many workflow paradigms. It doesn’t simplify agreeing on one. |