Paper DS16

From XPT to JSON for submissions: Are we ready for the change?

Poorna Suresh, Sycamore Informatics, India
Bill Qubeck, Sycamore Informatics, USA

ABSTRACT

Dataset-JSON, a structured, machine-readable format, is being considered as a long-term replacement for the legacy SAS XPT format for regulatory submissions. For sponsors, CROs, and technology providers, this signals a change that will require new tools, revised processes, and a deeper focus on metadata governance and validation strategies. This session explores why Dataset-JSON matters, what it offers, and how organizations should prepare. Building on insights from the recent PHUSE/CDISC pilot, we will examine Dataset-JSON's key attributes, including support for complex metadata, native compatibility with modern programming languages (R and Python), and its potential to enable real-time validation and automation. We will also address critical challenges, such as integration with legacy systems, and validation under GxP constraints. With the FDA actively seeking community input, now is the time for industry stakeholders to weigh in, prepare internally, and contribute to shaping the standards that will define the future of clinical data submissions. The evolution of data exchange standards in clinical research has transitioned from a fragmented, paper-bound world to a highly integrated, digital ecosystem. This shift has been driven by the dual needs for regulatory compliance and technological efficiency.

1. INTRODUCTION

The evolution of clinical data management has moved from physical stacks of paper to seamless, real-time digital integration. This journey highlights a shift from manual labor to global standardization, drastically improving the speed and accuracy of medical research.

The Era of Paper and Manual Records (Pre-1990s): Before the digital revolution, clinical data management was a cumbersome, entirely manual process. Researchers relied on paper Case Report Forms (CRFs) to record patient information by hand. In this era, data exchange was a literal term, requiring the physical mailing or faxing of massive binders of documents to sponsors and regulatory bodies. This method was notoriously slow and plagued by human transcription errors. Furthermore, the lack of a standardized language made it nearly impossible to aggregate or compare data across different trials, leaving much of the information siloed in paper archives.

The Digital Shift: Early EDC and HL7 (1990s): The late 20th century marked the birth of Electronic Data Capture (EDC) systems, which began replacing physical forms with digital databases. Around the same time, Health Level Seven (HL7) emerged to standardize how electronic health records (EHRs) exchanged information. While HL7 was primarily focused on healthcare administration rather than clinical research, it laid the essential groundwork for interoperability. However, the period was still defined by inconsistency; early EDC systems were "siloed," with each vendor using proprietary formats that prevented different software systems from communicating with one another.

The CDISC Revolution (2000s–2010s): The formation of the Clinical Data Interchange Standards Consortium (CDISC) in 1997 addressed the problems in clinical research for creating a unified language for the industry. Key milestones included the SDTM (Study Data Tabulation Model) for structuring raw submission data, ADaM (Analysis Data Model) for statistical analysis, and CDASH (Clinical Data Acquisition Standards Harmonization) for standardizing the questions asked on CRFs. The impact of these standards became absolute in 2016, when the U.S. FDA made CDISC compliance mandatory for all electronic submissions, forcing a global shift toward data uniformity.

Modern Interoperability: FHIR and Real-World Data (2020s–Present): Today, the industry has moved beyond just submitting data to integrate it from a vast array of sources. HL7 FHIR (Fast Healthcare Interoperability Resources) uses modern API technology to allow hospital records and trial databases to talk to each other in real-time, pulling patient data automatically and reducing manual entry. Simultaneously, standards like OMOP are being used to harness Real-World Data (RWD) from

wearables and insurance claims. This allows researchers to study how drugs perform in the real world, far beyond the controlled environment of traditional clinical settings.

2. DATASET-JSON: A MODERN SOLUTION FOR DATA EXCHANGE

JavaScript Object Notation (JSON) is a language-independent, text-based data serialization format. It serves as a lightweight alternative to more rigid or proprietary formats (e.g., SAS XPT) for the exchange of structured data across heterogeneous systems.

2.1 JSON mechanics

JSON (JavaScript Object Notation) is the standard for data exchange on the web because it is lightweight, language-independent, and easy for both humans and machines to read.

The mechanics that govern the structure and processing of JSON datasets are as follows.

- The Building Blocks (Data Types): JSON is built on two primary structures: a collection of name/value pairs (Objects) and an ordered list of values (Arrays). Within those, you can use these specific data types:
    - Strings: Must be wrapped in double quotes (e.g., "name" or "Bob").
    - Numbers: Integers or floating points (e.g., 25 or 3.14).
    - Booleans: Lowercase true or false.
    - Null: Represents an empty value (null).
    - Objects: Collections of key-value pairs wrapped in curly braces {}.
    - Arrays: Lists of values wrapped in square brackets [ ]
- Structural Rules: To be well-formed a JSON dataset must follow strict mechanical rules. If these are broken, a parser will throw an error:
    - Keys must be strings: Every key in an object must be enclosed in double quotes.
    - Colons and Commas: A colon (:) separates a key from its value. A comma (,) separates pairs within an object or elements within an array.
    - No trailing commas: Unlike some programming languages (like Python or modern JS), the last item in a JSON list or object cannot have a comma after it.
    - Nesting: Objects can contain arrays, and arrays can contain objects. This allows for complex, hierarchical data.
- Common Dataset Patterns: In data science and web development, you will usually see JSON formatted in one of two ways:

Standard JSON Array: A single file containing one large array of objects.

```
[
 {"id":1, "status": "active"},
 {"id":2, "status": "pending"}
]
```

JSON Lines (JSONL): Used for massive datasets. Each line is a separate, valid JSON object. This allows machines to process the file line-by-line without loading the entire dataset into memory (RAM).

Plaintext

```
{"id":1, "status": "active"}
{"id":2, "status": "pending"}
```

- JSON mechanics

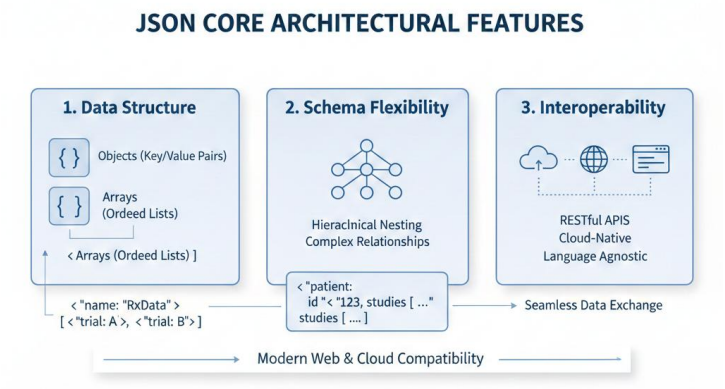
The journey of a JSON dataset typically follows this lifecycle:

  - Serialization: A program converts an internal data structure (like a Python dictionary) into a JSON string.
  - Transmission: The string is sent over a network (usually via an API).
  - Parsing (Deserialization): The receiving program "parses" the string, turning it back into a usable object in its own language.

Feature	JSON Requirement
Quotes	Double quotes only (")
Comments	Not supported (strictly data)
File Extension	.json
Encoding	Usually UTF-8

2.2. Core Architectural Features

- Data Structure: JSON is predicated on two universal data structures. a collection of name/value pairs (objects) and an ordered list of values (arrays).
- Schema Flexibility: Unlike fixed-width or legacy binary formats, JSON supports hierarchical nesting, allowing for the representation of complex relational data and multifaceted metadata within a single document.
- Interoperability: As a text-based standard, it is natively compatible with modern web protocols (REST/HTTP) and cloud-native architectures, ensuring seamless data parsing across diverse programming environments.



Due to its extensibility and efficiency, JSON is increasingly being adopted in clinical research. It is currently being evaluated as a robust transport format for regulatory data submissions, facilitating more agile data pipelines compared to traditional file-based standards.

2.3 The Metadata Hierarchy

It is vital to recognize that within the CDISC ecosystem, Define-XML remains the definitive regulatory source of truth for study metadata. While Dataset-JSON represents a significant leap in data exchange efficiency, it is designed to carry only a specific subset of the metadata required for a full regulatory submission. To ensure compliance and data integrity, it is helpful to view the relationship between the two formats as follows:

- **Define-XML (The Master Record):** Contains the comprehensive metadata "blueprint." This includes codelists, value-level metadata, derivation logic, origin information, and method descriptions that are essential for reviewers to understand the study.
- **Dataset-JSON (The Data Transport):** Focuses primarily on the data payload and the immediate structural metadata (item names and types) necessary to process the file. It does not—and is not intended to—replicate the rich, descriptive context housed in the Define-XML.

Key Distinctions

Feature	Define-XML	Dataset-JSON
Regulatory Status	The **Authoritative Source** for metadata.	A technical transport format for data.
Metadata Scope	**Comprehensive:** Includes derivations, comments, and links to documents.	**Minimal:** Includes only what is needed to describe the record structure.
Traceability	Provides the full path from raw data to analysis.	Relies on the Define-XML for external context and traceability.

Critical Reminder: In any regulatory submission, the Dataset-JSON files must be validated against the Define-XML to ensure consistency. If a discrepancy exists, the Define-XML is the version that governs the submission.

2.4. Strategic Value of JSON in Modern Data Architecture

Choosing JSON for modern data pipelines optimizes both technical interoperability and the execution of core business logic.

- **Leveraging JSON for Modern Web Services:** JSON is the backbone of the modern web, primarily due to its native compatibility with JavaScript and the Document Object Model (DOM).
    - Outcome: It eliminates the overhead of complex parsing logic required by XML.
    - Function: By using a text-based format that maps directly to programming language objects, developers can achieve faster serialization and deserialization, reducing communication latency.
- **The Role of JSON in Scalable Data Exchange:** In cloud-native environments, systems must handle massive throughput with minimal resource consumption.
    - Outcome: High-concurrency performance and reduced bandwidth costs.
    - Function: JSON’s minimalistic syntax (low metadata-to-payload ratio) ensures that data packets remain small. This efficiency allows distributed microservices to synchronize across global networks without the heavy lifting associated with legacy binary protocols.
- **Streamlining Regulatory Submissions with JSON:** In highly regulated fields like clinical research, the transition from fixed width (SAS XPT) to JSON represents a move toward data integrity and transparency.
    - Outcome: Improved audit trails and easier validation of complex clinical datasets.

- Function: JSON facilitates self-describing data structures. By nesting metadata directly alongside clinical observations, researchers ensure that the context of the data is never lost during transport between sponsors and regulatory bodies (e.g., FDA/EMA).

- **Enabling Real-Time Data Interoperability:** Modern applications require data to flow seamlessly between disparate platforms (e.g., a mobile app, a Python-based AI model, and an SQL database).
  - Outcome: Language-agnostic integration.
  - Function: JSON acts as a universal translator. As almost every modern programming language includes a native JSON parser, it removes the silo effect, allowing a front-end interface to consume the same data structure that a back-end analytical engine processes.

3. LIMITATIONS OF SAS XPT

The SAS Version 5 Transport File (XPT) is a non-proprietary, open-specification binary format developed in the 1980s. Its primary purpose was to enable the transfer of datasets between different operating systems (e.g., from a mainframe to a PC) without losing numerical precision.

Comparison of SAS XPT (Version 5) and JSON is listed in the table below

Feature	SAS XPT (Version 5)	Dataset-JSON (CDISC Standard)
Technical Format	Binary (Fixed-width blocks)	Text-based (UTF-8 encoding)
Variable Names	Limited to 8 characters	Unlimited (supports descriptive names)
Labels & Length	Labels: 40 characters; Strings: 200 characters	Unlimited (handles long text/comments)
Readability	Requires specific tools (SAS, R, Python)	Human-readable; opens in any text editor
Metadata	Decoupled (requires separate Define-XML)	Linked/Integrated (metadata can be embedded)
Scalability	Tabular only (Rows/Columns)	Supports Hierarchical/Nested data
Regulatory Status	Current mandatory standard	Under FDA Pilot/Evaluation (2025–2026)

3.1 Why the Shift to Dataset-JSON Matters in Clinical Trials

The shift towards a modern technique matters for the reasons discussed below –

- **Data Integrity and Self-Description:** In SAS XPT, the data is separated from its description. If you lose the accompanying define.xml file, you might see a variable named LBTESTCD and not know it stands for Laboratory Test Code. JSON allows the metadata to travel with the data more closely. Even in a simple text viewer, the key-value pair "LaboratoryTestCode": "Glucose" is immediately understandable to a reviewer.
- **Handling Complex Relationships:** Clinical data is becoming more complex (e.g., genomics, digital health sensor data). SAS XPT forces this data into flat tables, often requiring multiple complicated merges. JSON naturally supports nesting. You can have a "Patient" object that contains an array of "Visits," which in turn contains an array of "Vital Signs," keeping the logical relationship intact without complex join keys.

- **Modern Tech Interoperability:** Most modern web APIs and cloud-based analytics (like those used in decentralized clinical trials) speak JSON natively. Using SAS XPT often requires an extra conversion step that increases the risk of data truncation or character encoding errors (since XPT v5 is limited to US ASCII).
- **File Size and Storage:** Contrary to common belief, JSON often results in smaller file sizes for clinical datasets. XPT allocates a fixed width for every row; if a column is defined as 200 characters but only contains 5, XPT still stores 195 empty spaces. JSON only stores the actual data present.

- **Modernizing the Clinical Data Pipeline:** Dataset-JSON represents the long-awaited evolution of clinical data transport, designed to replace the decades-old SAS Version 5 (XPT) format. While XPT has been the industry standard since the 1980s, its rigid technical limitations such as 8-character variable names and restricted character lengths no longer meet the needs of modern, data-rich clinical trials. Dataset-JSON removes these technical handcuffs by leveraging a text-based, open-standard format that is natively understood by nearly every modern programming language, including R, Python, and Java. This transition allows the industry to move away from proprietary dependencies and toward a more flexible, interoperable data ecosystem.

- **Efficiency and Regulatory Alignment:** For stakeholders concerned with speed and compliance, Dataset-JSON offers significant operational advantages. Its architecture is optimized for the cloud, supporting faster data transfers, reduced storage footprints, and seamless integration with REST APIs. This paves the way for real-time data monitoring and rolling reviews with regulatory bodies. As of today, major agencies like the FDA have moved beyond successful pilot programs and are actively integrating Dataset-JSON into their submission frameworks. By adopting this format, sponsors can ensure their data is future-proofed for upcoming versions of CDISC standards that will finally break the restrictive legacy limits of the XPT era.

- **Digital Transformation:** Ultimately, the shift to Dataset-JSON is not merely a change in file extension. It is a foundational step toward the digital transformation of clinical research. By adopting a format that supports UTF-8 encoding (enabling global character sets) and unlimited metadata labels, organizations can improve the quality and clarity of the data sent to reviewers. While the industry is currently in a dual-support phase, early adoption of Dataset-JSON positions a company as a leader in innovation, reducing the long-term risk of technical debt and ensuring that clinical assets remain accessible and actionable for years to come.

4. LEVERAGING SAS FOR DATASET-JSON PROCESSING

In clinical research and data science industries, the transition from the traditional SAS Transport File (XPT) format to Dataset-JSON is a major shift toward modernized, web-ready data exchange. Dataset-JSON is designed to be more flexible, lightweight, and compatible with modern programming languages while maintaining the integrity of CDISC standards.

Dataset-JSON (specifically CDISC Dataset-JSON) contains a columns section for metadata and an itemGroupData section for the actual observations. A simple libname statement might not be enough for complex automation.

4.1. Strategy for Clinical Data

- Extract Metadata: Use the JSON engine to read the columns array to determine lengths, types, and labels.
- Dynamic Code Generation: Use SAS macros to generate a DATA step that applies the correct attributes to the variables.
- Data Loading: Read the itemGroupData and map it back to the structured variables.

4.2. Recommended Practices

- Using File References: Always use filename statements to manage your JSON paths; it makes the code more portable across environments (Windows vs. Unix).
- Validate against Schema: Since JSON doesn't have the rigid structure of a SAS7BDAT, a JSON schema validator (often via a Groovy script or a Java-based transformation within SAS) ensures the file meets CDISC standards.
- Memory Management: For extremely large JSON files, the libname JSON engine can be memory intensive. In those cases consider using the DS2 language within SAS which offers more robust parsing capabilities.

5. DATASET-JSON EXAMPLE ANNOTATED WITH MAPPING TO DEFINE-XML

When mapping a Dataset JSON (part of the CDISC Dataset-JSON standard) to Define-XML one is essentially linking the raw data structure to its formal metadata definition.The Define-XML file acts as the lookup table that explains what the variables in the JSON actually mean.

Below is an example of a Demographics (DM) domain entry in Dataset-JSON format, annotated with how each part relates to the Define-XML standard.

JSON

```
{
  "creationDateTime": "2026-01-15T09:13:00",
  "datasetJSONVersion": "1.1",
  "itemGroupData": {
    "IG.DM": {  // Maps to Define-XML: ItemGroupDef/@OID
      "records": 1,
      "name": "DM", // Maps to ItemGroupDef/@Name
      "label": "Demographics", // Maps to ItemGroupDef/Description

      "items": [ // Maps to ItemGroupDef/ItemRef (The ordered list of
variables)
        {
          "OID": "IT.DM.STUDYID",
          "name": "STUDYID",
          "label": "Study Identifier",
          "type": "string"
        },
        {
          "OID": "IT.DM.USUBJID",
          "name": "USUBJID",
          "label": "Unique Subject Identifier",
          "type": "string"
        },
        {
          "OID": "IT.DM.AGE",
          "name": "AGE",
          "label": "Age",
          "type": "integer"
        }
      ],

      "itemData": [ // The actual data rows
        [
          "PROJECT-XYZ", // Value for ItemRef (Order 1)
          "001-01",      // Value for ItemRef (Order 2)
          45             // Value for ItemRef (Order 3)
        ]
      ]
    }
  }
```

}

Key Mapping Components

Dataset JSON Element	Define-XML Equivalent	Purpose
itemGroupData Key	ItemGroupDef/@OID	The unique identifier for the dataset (e.g., IG.DM).
name	ItemGroupDef/@Name	The short name of the domain (e.g., DM).
label	ItemGroupDef/Description	The human-readable description of the dataset.
items[].OID	ItemDef/@OID	Links the data column to its specific metadata definition.
items[].type	ItemDef/@DataType	Ensures the data type (string, integer, float) matches the metadata.
itemData	N/A	This is the actual subject data; Define-XML describes its *structure*, not these values.

Why does this mapping matter?

The OID (Object Identifier) is the most critical link. When a regulatory reviewer opens your data, their software uses the OID in the JSON to find the corresponding ItemDef in the define.xml. This tells them, for example, that the value 45 in the third column represents AGE and is measured in YEARS.

6. METADATA MANAGEMENT IN DATASET-JSON

In the Dataset-JSON framework, metadata is handled through a layered approach. Basic structural metadata is embedded directly within the JSON file, while comprehensive, study-level metadata remains linked to an external Define-XML document.

6.1. Metadata Architecture

Dataset-JSON (specifically version 1.1) follows a data-plus-metadata structure. It is designed to be semi-self-describing, i.e, one can understand the basic structure of the table without needing external files, but it relies on Define-XML for the full regulatory context.

- File-Level Metadata: At the top level, the JSON object contains attributes that describe the file's origin and versioning.
  - datasetJSONVersion: Specifies the version of the standard (e.g., "1.1").
  - fileOID: A unique identifier for the specific file.
  - asOfDateTime: The timestamp when the source data was extracted.
  - originator: The organization that generated the file.
  - sourceSystem/sourceSystemVersion: Details about the software or database used to create the dataset.
- Dataset-Level Metadata: This section identifies the CDISC domain or analysis dataset:
  - studyOID & metaDataVersionOID: These act as foreign keys that must match the identifiers used in the associated Define-XML file.
  - itemGroupOID: Corresponds to the dataset's OID in Define-XML (e.g., IG.DM for Demographics).

- label: The descriptive label for the dataset.

- Column-Level Metadata (items array): Dataset-JSON includes an array called items that defines every variable (column) in the dataset. This is a significant upgrade over XPORT because it supports modern data types.
  - itemOID: Unique identifier for the variable.
  - name: The variable name (no longer restricted to 8 characters).
  - label: Variable description (no longer restricted to 40 characters).
  - dataType: Supports string, integer, decimal, float, double, boolean, date, datetime, and time.
  - targetDataType: Used when the data needs to be converted for a specific system (e.g., telling a receiver that an ISO 8601 string should be treated as a SAS numeric date).

6.2. The Linkage with Define-XML

While Dataset-JSON contains enough metadata to render a table, Define-XML remains the source of truth for regulatory submissions.

Feature	Contained in Dataset-JSON?	Contained in Define-XML?
Variable Names & Labels	Yes	Yes
Data Types	Yes (JSON-specific)	Yes (Standard-specific)
Codelists / Controlled Terminology	No	Yes
Value-Level Metadata	No	Yes
Origin / Derivations	No	Yes
Method / Comments	No	Yes

6.3. Key Improvements to Legacy Formats

- Extended Lengths: Variable names can exceed 8 characters, and labels can exceed 40.
- Unicode Support: Unlike XPORT (ASCII), Dataset-JSON supports UTF-8, allowing for international characters in metadata and data.
- Type Safety: Explicitly defines if a field is a Boolean or a high-precision decimal, reducing ambiguity during data ingestion.

7. VALIDATION IN CDISC DATASET-JSON

7.1 Parallel Submission Pipeline - Proposed

Implementing a Parallel Submission Pipeline (PSP) over the next 12 to 24 months represents a shift from the traditional waterfall approach where data is cleaned, locked, and then analyzed to a continuous, real-time flow of data toward regulatory and internal milestones.

This model is designed to compress drug development timelines by 15–25% by overlapping data management, medical writing, and regulatory operations. The PSP model moves away from sequential dependencies. Instead of waiting for a Database Lock (DBL), the pipeline operates on Continuous Data Integration.

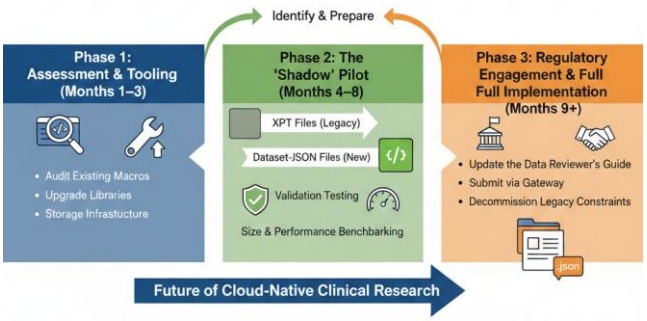
The Three Concurrent Workstreams

- Continuous Data Stream: Clinical data is ingested, cleaned, and SDTM-converted in monthly or event-driven batches rather than at the end of the study.

- Iterative Shelling & Authoring: Medical writers begin drafting Module 2 (Summaries) and Module 5 (CSRs) using dry-run data or interim cuts, rather than waiting for the final Tables, Listings, and Figures (TLFs).
- Simultaneous Global Filings: Instead of filing with the FDA and waiting 3–6 months for the EMA/PMDA, the organization maintains a Global Master Dossier that allows for near-simultaneous submission across multiple health authorities (HAs).

7.2. Suggested timeline

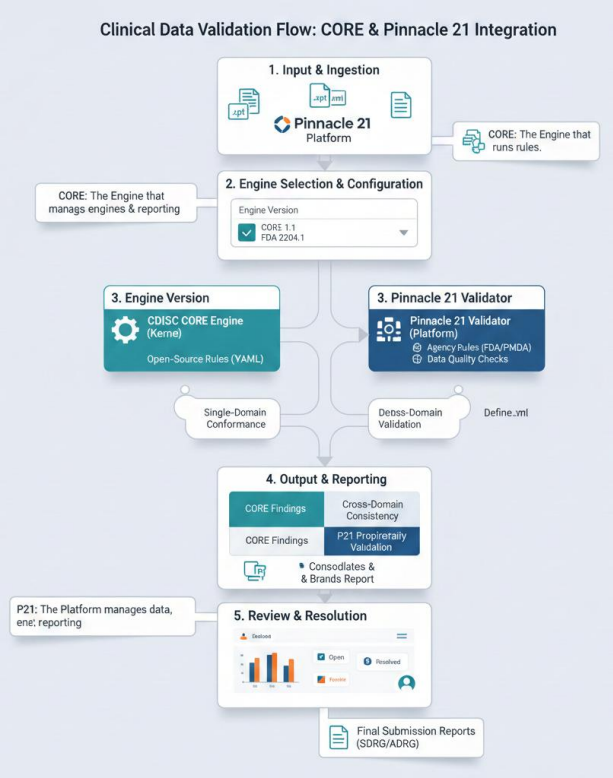
- **Phase 1: Assessment & Tooling (Months 1–3):** The first step is to inventory your current clinical data pipeline. Most legacy systems are hard-wired for XPT.
    - Audit Existing Macros: Identify SAS macros or R scripts that explicitly use libname xport or write_xpt.
    - Upgrade Libraries: Ensure your environment has the latest CDISC-compliant libraries (e.g., the cdisc-dataset-json Python library or the sdtm.oak framework in R).
    - Storage Infrastructure: Verify that your Clinical Data Repository (CDR) can ingest and index .json or .ndjson files as first-class citizens alongside legacy formats.
- **Phase 2: The Shadow Pilot (Months 4–8):** Before committing to a live submission, run a Shadow Pilot using data from a completed Phase 2 or Phase 3 study.
    - Parallel Generation: Generate both XPT and Dataset-JSON files from the same source data.
    - Validation Testing: Run the datasets through a validator (like Pinnacle 21 Enterprise or the Community version) that supports Dataset-JSON to ensure metadata consistency with your Define-XML 2.1.
    - Size & Performance Benchmarking: Compare the file sizes and read/write speeds. You will likely see significant efficiency gains in large datasets (e.g., Laboratory or ECG findings).
- **Phase 3: Regulatory Engagement & Full Implementation (Months 9+):** Once your internal tools are validated, one can move towards the first official submission.
    - Update the Data Reviewer's Guide (cDRG/adrg): Explicitly document the use of Dataset-JSON in the Data Transmission section of your reviewer's guide so the FDA/PMDA reviewers are prepared.
    - Submit via Gateway: Use the standard Electronic Submissions Gateway (ESG). Ensure that the folder structure follows the eCTD specifications, simply replacing .xpt files with .json files in the m5 folder.
    - Decommission Legacy Constraints: Begin designing new studies that utilize variable names longer than 8 characters and labels longer than 40 characters, fully leveraging the new format's capabilities.



7.3 A validation flow showing CORE and Pinnacle 21

To visualize the relationship between CDISC CORE and Pinnacle 21, it is helpful to think of it as a sandwich. CDISC CORE provides the official open-source engine and rules, while Pinnacle 21 acts as the professional wrapper that provides the user interface, additional agency checks (FDA/PMDA), and final submission reporting.

Validation Flow Diagram



8. VERIFICATION IN CDISC DATASET-JSON

Verification for CDISC Dataset-JSON is critical as the industry transitions from the legacy SAS V5 Transport (.xpt) format to this modern, JSON-based exchange standard.

8.1. Technical Verification Steps

Unlike XPT files, Dataset-JSON is a text-based format that requires specific technical checks:

- Schema Validation: Use a JSON Schema validator to ensure the file follows the official CDISC schema. This detects structural errors (e.g., missing commas, incorrect data types for metadata fields) that would break automated loaders.
- Encoding Check: Verify that the file is encoded in UTF-8. This is a major advantage over XPT, as it allows special characters and international symbols, but it must be strictly enforced.
- Precision and Data Types: JSON treats numbers and strings differently. QC must ensure that numeric precision (especially for floating-point numbers) is maintained and that dates follow ISO 8601 format.
- File Size & Integrity: For large datasets, verify that the file isn't truncated. CDISC is also introducing NDJSON (Newline Delimited JSON) for extremely large datasets to facilitate streaming data during QC.

8.2. Verification of Metadata Linkage

A core requirement of Dataset-JSON is its relationship with Define-XML. The JSON file itself contains item metadata (labels, types), but the source of truth still remains the Define file as mentioned above.

- OID Matching: The itemGroupOID in the JSON must match the ItemGroupDef OID in the Define-XML.
- Variable Order: The order of columns in the columns array of the JSON should match the order defined in the metadata to prevent mapping errors during review.

8.3. Using CDISC Dataset-JSON Technique

In clinical trials, one is likely comparing Dataset-JSON (the new CDISC standard) against legacy XPT files.

- dsjconvert (CLI Tool): This is a specialized Python tool used to convert between XPT and Dataset-JSON v1.1. You can convert the XPT to JSON and then use a standard JSON diff tool.
- Lex Jansen's SAS Macros: For those staying within the SAS environment, Lex Jansen provides open-source macros that can read Dataset-JSON into SAS datasets, allowing you to use PROC COMPARE.

8.4. Common Pitfalls in Dataset-JSON

During the FDA/PHUSE pilots, several common issues were identified that should be included in the checklist:

- Mismatch in OIDs: The OID in the JSON header doesn't match the OID in the Define-XML.
- Date Formatting: Dates appearing as strings in some files and numeric timestamps in others.
- Null Handling: Improper representation of "missing" data (e.g., using an empty string "" instead of a JSON null).
- Large File Performance: Challenges in opening 5GB+ JSON files in standard text editors require specialized viewers or streaming tools.

9. QUALITY CONTROL IN CDISC DATASET-JSON

9.1. The Multi-Layered QC Framework

Quality Control for Dataset-JSON is typically divided into three distinct layers to ensure the data is fit for use by regulatory agencies like the FDA.

Layer	Focus Area	Key Verification Points
Technical/Schema	JSON Structure	Compliance with the CDISC Dataset-JSON Schema (e.g., v1.1). Checks for required fields like itemGroupData, creationDateTime, and datasetJSONVersion.
Metadata Integrity	Define-XML Alignment	Ensuring the OIDs (Object Identifiers) and metadata in the JSON file match exactly with the corresponding Define-XML file.
Content/Conformance	CDISC Standards	Traditional SDTM/ADaM/SEND conformance rules (e.g., variable lengths, controlled terminology, and cross-domain consistency).

9.2. Tools for Quality Control

As of today, the ecosystem for Dataset-JSON QC has matured significantly:

- CDISC CORE (Conformance Rules Engine): The primary open-source tool for executing CDISC conformance rules directly against Dataset-JSON files.
- Pinnacle 21/Beacon: Commercial validators have added native support to read and validate Dataset-JSON in the same way they process XPT.
- Programming Libraries:
    - **R Ecosystem: The Lead in Innovation**
        - The R community, primarily through the Pharmaverse initiative, has the most user-friendly tools for converting data frames directly into validated Dataset-JSON.
        - {datasetjson} (by Atorus Research):
            - Status: The industry standard for R.
            - Key Features: It allows you to take a standard R data frame and "wrap" it with the necessary CDISC metadata (Study OID, ItemGroup OID, etc.). It supports both reading and writing v1.1 files.
            - Benefit: It automatically handles the conversion of R factors and dates into the ISO 8601 strings required by the CDISC standard.
        - {sdtm.oak}:
            - Status: An EDC-agnostic SDTM generation engine.
            - Key Features: While its primary job is mapping raw data to SDTM, it is increasingly being integrated with {datasetjson} to provide a single-click pipeline from raw data to a submitted JSON file.
    - **Python Ecosystem: The Scalability Choice**
        - Python is favored for high-performance data engineering, particularly when dealing with "Big Data" (e.g., biomarkers or sensor data) where XPT files become unwieldy.
        - dsjconvert (by Sam Hume/CDISC contributors):
            - Status: A dedicated CLI tool and Python package.
            - Key Features: Specifically built for bidirectional conversion. It can take an existing .xpt file and a define.xml and produce a fully compliant .json or .ndjson file.
            - Benefit: It includes a "no-define" mode that infers metadata from the dataset itself, which is useful for internal data exploration before formal submission.
        - cdisc-dataset-json (GitHub):
            - Status: The reference implementation.
            - Key Features: Primarily used for validating files against the official JSON schemas. It is often used as a backend for larger Python-based clinical data repositories (CDRs).

- **SAS Ecosystem: The Legacy Bridge**
  - Since SAS is still the primary environment for most statistical programmers, the tools here are designed to bridge the gap between classic PROC steps and modern JSON output.
  - PROC JSON:
    - Status: Native SAS procedure.
    - Usage: You can use PROC JSON with the EXPORT statement to create JSON files. However, creating a *CDISC-compliant* Dataset-JSON requires a specific "nesting" of metadata that PROC JSON does not do automatically.
  - The Lex Jansen Macros:
    - Status: Widely adopted open-source macros.
    - Key Features: Developed by Lex Jansen (a key CDISC architect), these macros automate the complex task of wrapping SAS datasets into the Dataset-JSON structure using PROC JSON and DATA steps.
    - Benefit: They include built-in comparison tools to verify that the generated JSON contains the exact same data values as the source SAS dataset.

- Direct Comparison via Python

  Python is the most flexible tool for this task because it has native libraries for both formats.

  - Step 1: Read the Files
    - Use pandas with pyreadstat to read the XPT file.
    - Use the standard json library to read the JSON file.
  - Step 2: Structural Alignment
    - Since JSON can be nested and XPT is flat, you must flatten the JSON into a table (DataFrame) using pd.json_normalize().
  - Step 3: Comparison
    - Use the datacompy library, which is specifically designed to compare two DataFrames and provide a human-readable report of differences in rows and columns.

    Python Code

```
import pandas as pd
import datacompy
#Load SAS XPT
Df_xpt = pd.read_sas('data.xpt', format = 'xport')
#Load and flatten JSON
Df_json = pd.read_json('data.json')
#Compare
Compare = datacompy.Compare(df_xpt, df_json, join_columns = 'ID_COLUMN')
print(compare.report())
```

9.3. Tool-Based Comparisons

If you prefer a UI or specialized software, consider these approaches:

Method	Best For	Recommended Tools
Visual Diff	Quick spot-checks	Beyond Compare or DeltaWalker. These can often handle XPT (via plugins) and JSON side-by-side.
Command Line	Automation	jq (for JSON) + csvdiff. Convert both to CSV first, then diff the CSVs.
Online	Non-sensitive data	JSONCompare or Diffchecker (after converting XPT to a text-based format).

9.4. Key Challenges

When comparing these two, one is likely to encounter the following false positives:

- Variable Lengths: XPT (V5) limits variable names to 8 characters and labels to 40 characters. JSON has no such limit, so names may be truncated in XPT.
- Data Types: JSON doesn't distinguish between a float and an integer as strictly as SAS. A "1" in JSON might be compared against "1.0" in SAS.
- Character Encoding: XPT is traditionally US-ASCII, while JSON is UTF-8. Special characters may break or change during conversion.
- Metadata: XPT stores labels and formats in the file header; JSON often stores this in a separate define.xml or within a specific metadata object in the JSON itself.

10. GxP APPROACH

In clinical research, a GxP approach (encompassing GCP, GLP, and GMP) ensures that every system, piece of equipment, and process is fit for its intended use and remains in a state of control throughout its lifecycle. This is achieved through the dual pillars of Qualification and Change Control.

10.1. The GxP Qualification Framework

Qualification is the formal process of proving that equipment or systems are properly installed, work correctly, and lead to expected results. In clinical settings, this often follows the V-Model or a risk-based approach like GAMP 5 (Good Automated Manufacturing Practice).

The Four Stages of Qualification

Stage	Definition	Key Activities
Design (DQ)	Verification that the proposed design is suitable for the intended purpose.	Reviewing User Requirement Specifications (URS).
Installation (IQ)	Verification that the system is installed according to the manufacturer's specifications.	Checking parts, wiring, software versions, and environmental conditions.

Operational (OQ)	Verification that the system operates as intended throughout all anticipated operating ranges.	Testing alarms, "worst-case" scenarios, and functional limits.
Performance (PQ)	Verification that the system performs consistently under real-world clinical conditions.	Testing the integrated system with actual data/samples over time.

10.2. Change Control: Maintaining the Validated State

Change control is a formal system where qualified representatives review proposed changes that might affect the validated status of a system. Its primary goal is to ensure that there is no change whether a software patch, a hardware upgrade, or a process shift that unintentionally compromises patient safety or data integrity.

Key Components of GxP Change Control

- Impact Assessment: Evaluation of how the change affects the "validated state." This must consider regulatory impact (e.g., does it require a new FDA filing?) and risk to data.
- Traceability: Every change must be documented in a way that is Attributable, Legible, Contemporaneous, Original, and Accurate (ALCOA+).
- Audit Trails: For computerized systems, changes must be recorded in secure, computer-generated, time-stamped audit trails that do not obscure previous entries (PharmTech, n.d.).
- Approval Workflow: Changes must be reviewed and approved by Quality Assurance (QA) and Subject Matter Experts (SMEs) before implementation.

10.3. Risk-Based Approaches and Emerging Trends

Modern GxP management is moving away from one-size-fits-all checklists toward a Quality Risk Management approach.

- Tiered Validation: For clinical biomarkers or laboratory equipment, a fit-for-purpose approach is often used, where the level of qualification depends on the risk the data poses to clinical decisions.
- Artificial Intelligence (AI): As AI is integrated into clinical monitoring, traditional change control is being challenged by adaptive algorithms. New frameworks for Explainable AI (XAI) are being developed to ensure these systems meet the same transparency and traceability standards as traditional systems.

11. CONCLUSION AND FUTURE DIRECTIONS

As the clinical research industry moves away from the decades old SAS Transport (XPT) format, Dataset-JSON has emerged as the cornerstone of modern data exchange. It is no longer just a pilot project, but a rapidly maturing standard designed to support the need for speed in regulatory reviews.

11.1. Regulatory Acceptance and the shift from XPT

The primary driver for Dataset-JSON is the removal of legacy technical debt inherent in XPT v5 (e.g., 8-character variable names, 200-character string limits, and lack of UTF-8 support).

- FDA and PMDA Readiness: Following successful pilots, the FDA is moving toward native consumption of Dataset-JSON. The future strategy involves updating the Data Standards Catalog to include Dataset-JSON as a supported alternative to XPT.
- Beyond Same Content, Different Suitcase: While initial use cases focused on simply mapping existing SDTM/ADaM data into JSON, the future focus is on leveraging JSON’s nested structure to include Audit Trails and Source Data (like EHR or Lab transfers) directly within the submission package.

11.2. Technical Evolution (Dataset-JSON v1.1 and beyond)

The release of Dataset-JSON v1.1 addresses critical findings from industry pilots to make the standard more robust for global use:

- Support for NDJSON (Newline Delimited JSON): To handle big data (e.g., high-frequency sensor data or large genomics datasets), the standard now supports NDJSON. This allows for streaming data line-by-line, preventing memory crashes associated with loading multi-gigabyte monolithic JSON files.
- Precision and Data Types: A new decimal data type has been introduced, where numbers are stored as strings to prevent the rounding errors common in standard JSON floating-point libraries.
- ISO 8601 Standardization: Moving away from software-specific date epochs (like SAS 1960 or R 1970), the future standard mandates ISO 8601 strings, ensuring total interoperability between different programming environments.

11.3. Integration with the CDISC 360 Ecosystem

Dataset-JSON is a key enabler for CDISC 360, the initiative for end-to-end metadata-driven automation.

- DDF and USDM Integration: The standard is being aligned with the Digital Data Flow (DDF) and the Unified Study Definition Model (USDM). This allows for the automatic generation of datasets directly from the digital protocol.
- API-First Exchange: Instead of physical file transfers, the Dataset-JSON API v1.0 enables real-time data pulls. Reviewers could query specific subsets of data (e.g., "all subjects with Systolic BP > 140") directly from a repository without downloading the entire study.
- CORE (CDISC Open Rules Engine): Future workflows will see Dataset-JSON files validated in real-time by CORE, providing instant feedback on conformance during the data conversion process rather than at the end of a study.

11.4. Readiness Checklist

This checklist is designed for a Sponsor lead or Clinical Data Manager preparing for the transition from traditional XPT-based submissions to the modern Dataset-JSON standard. Moving to Dataset-JSON isn't just a file format swap; it’s a shift toward a more metadata-driven ecosystem. Here is a readiness checklist structured for execution.

- Standards & Metadata Governance: Before a single file is generated, your structural rules must be locked down. Unlike XPT, Dataset-JSON relies heavily on the Define-XML as the source of truth for the data structure.
    - Define XML Linkage Rules: Establish a formal process for ensuring the ItemGroupData and ItemData in the JSON file map 1:1 with the ItemDef and ItemGroupDef in your Define-XML.
    - OID Governance: Standardize the generation of Object Identifiers (OIDs). These must be unique, persistent, and synchronized across your clinical metadata repository (MDR).
    - External Terminology Management: Document the workflow for Controlled Terminology (CT). Since JSON files carry the data but not the full dictionary metadata, verify that your Define-XML correctly references the specific CDISC CT versions.
    - Dataset-JSON Versioning: Confirm which version of the CDISC Dataset-JSON API/Standard is being used (e.g., v1.1) and ensure it matches the metadata schema.
- Conversion Strategy: The FDA is in a transitional phase regarding Dataset-JSON. Your strategy must mitigate the risk of technical rejection while maintaining data integrity.
    - System of Record (SoR) Designation: Decide if your SAS datasets (XPT) remain the legal system of record or if the JSON files take precedence during the internal QC cycle.

- Parallel Generation Workflow:
  - *Option A:* Generate XPT, then convert XPT to JSON.
  - *Option B:* Generate both simultaneously from a common source (e.g., an R or Python data frame).
- Traceability Audit: If generating in parallel, implement a round-trip test to ensure that converting a JSON back to XPT results in zero data loss or precision changes.
- Submission Packaging: Verify the folder structure required for the eCTD (Electronic Common Technical Document) to ensure JSON files are placed in the correct m5 subfolders.

- Validation & QC: Standard Pinnacle 21 or software-specific checks for XPT are no longer sufficient. You need a three-layer validation approach.
  - Schema Validation: Use a JSON Schema validator to ensure the file structure follows the CDISC technical specification (e.g., correct use of arrays, strings, and null values).
  - Conformance Rules Execution: Run CDISC-standard validation rules (like SDTM or ADaM conformance) specifically against the JSON content, not just the source data.
  - Define-XML Consistency Check: Ensure every variable present in the JSON is defined in the XML. Validate that data types (integer vs. float) in the JSON align with the DataType attribute in the Define-XML.
  - Visual Audit: Use a Dataset-JSON viewer or a transformation tool to render the JSON into a human-readable table for manual medical monitoring and lead programmer review.

11.5. Open-Source and AI Powering Implementation

The future of these standards is heavily reliant on the CDISC Open-Source Alliance (COSA).

- Automated Mapping: AI and Machine Learning tools are being developed to automate the mapping of raw data into Dataset-JSON, reducing the manual effort currently required for SDTM/ADaM conversion.
- Viewer Tools: Since JSON is not naturally human-readable in its raw form, a major future direction is the development of standardized, open-source Dataset-JSON Viewers that provide a tabular interface for medical reviewers.

REFERENCES

1. FDA History: *FDA and Clinical Drug Trials: A Short History.* This document outlines the 1962 Drug Amendments which first mandated "adequate and well-controlled" studies, creating the initial need for organized data collection. [Source: FDA.gov]
2. The Rise of EDC: *The Importance of EDC Systems in Supporting Early Phase Clinical Trials.* Medidata (2022). Discusses how EDC systems replaced paper to streamline data flows in the late 20th century. [Source: Medidata Blog]
3. The Mandatory Shift: *Providing Regulatory Submissions in Electronic Format — Standardized Study Data.* U.S. Food and Drug Administration (2014/2016). This binding guidance made CDISC (SDTM/ADaM) mandatory for NDAs and BLAs. [Source: FDA Guidance]
4. SDTM Evolution: *Evolution and implementation of the CDISC Study Data Tabulation Model (SDTM).* ResearchGate (2008). Traces the early development of the SDTM framework. [Source: ResearchGate]
5. Impact Metrics: *Understanding CDISC Standards: A Complete Guide.* Quanticate. Notes that implementing CDISC from the start can reduce resources needed for research by up to 60%. [Source: Quanticate]

6. FHIR to CDISC Mapping: *The Story Behind the HL7 FHIR to CDISC Mapping Implementation Guide.* Journal of the Society for Clinical Data Management (2022). Explains the technical bridge between healthcare APIs (FHIR) and research standards (CDASH/SDTM). [Source: JSCDM]

7. OMOP Common Data Model: *Standardized Data: The OMOP Common Data Model.* OHDSI.org. Describes the framework used to standardize observational and real-world data (RWD) for systematic analysis. [Source: OHDSI]

8. Real-World Evidence (RWE): *Use of Fast Healthcare Interoperability Resources (FHIR) in the Generation of Real World Evidence (RWE).* CDISC Knowledge Base. A pilot study demonstrating how EHR data is pulled via FHIR to support regulatory applications. [Source: CDISC KB]

9. CDISC Official Site: Foundational standards (SDTM, ADaM, CDASH).

10. HL7 International: Information on FHIR resources and healthcare interoperability.

11. OHDSI (Observational Health Data Sciences and Informatics): For OMOP and large-scale observational research.

12. Dataset-JSON v1.1 Specification (Released Dec 2024): The primary technical manual detailing the structure, metadata, and data types for clinical datasets. [View Dataset-JSON v1.1 Standard]

13. Dataset-JSON User Guide: Practical implementation guidance for sponsors and developers. [CDISC Wiki: User's Guide]

14. Dataset-JSON API v1.0: Specification for API-based data exchange, facilitating real-time data access. [API Standard Specification]

15. White Paper WP-88: *"Dataset-JSON as an Alternative Transport Format for Regulatory Submissions: Final Pilot Report"* (June 2024).[Read the Final Pilot Report (PDF)]

16. PHUSE Advance Hub: Ongoing project documentation for the *Optimizing the Use of Data Standards* working group. [PHUSE Dataset-JSON Project Page]

17. Federal Register Notice (FDA-2025-N-0129): *"Electronic Study Data Submission; Request for Comments on Dataset-JSON v1.1."* [View Official FDA Notice (April 2025)]

18. Lex Jansen's SAS Tools: A widely cited set of macros and tools for converting SAS datasets to Dataset-JSON. [PharmaSUG Paper: Working with Dataset-JSON using SAS]

19. Agilent. (2014). *Harmonizing USP <1058> and GAMP for analytical instrument qualification*. https://www.agilent.com/Library/slidepresentation/Public/Harmonizing_USP_1058_GAMP_Pharmaceutical_Engineering.pdf

20. Clinical Medicine and Health Research Journal. (2025). *Explainable AI in GxP validation: Balancing automation, traceability, and regulatory trust in the pharmaceutical industry*. http://www.cmhrj.com/index.php/cmhrj/article/download/509/320

21. PharmTech. (n.d.). *Configuring software for compliance with 21 CFR Part 11 audit trail requirements*. https://www.pharmtech.com/view/configuring-software-compliance-21-cfr-part-11-audit-trail-requirements

22. Todde, S. (2017). Guidance on validation and qualification of processes and operations involving radiopharmaceuticals. *EJNMMI Radiopharmacy and Chemistry*, *2*(1). https://doi.org/10.1186/s41181-017-0025-9

23. Wang, T. (2017). Ensuring quality and compliance in outsourcing of bioanalysis of clinical biomarkers. *Bioanalysis*, *9*(6), 501–504. https://doi.org/10.4155/bio-2017-0008

24. World Health Organization (WHO). (2019). *Guideline on data integrity*. https://iris.who.int/bitstream/handle/10665/330818/DI334-773-793-eng.pdf

25. World Journal of Pharmaceutical Research. (n.d.). *Change control in pharmaceutical industry*. https://wjpr.s3.ap-south-1.amazonaws.com/article_issue/1451557297.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Poorna Suresh
Company: Sycamore Informatics
Address: Bangalore, India
Email: poorna.suresh@sycamoreinformatics.com

Author Name: Bill Qubeck
Company: Sycamore Informatics
Address: Waltham, MA, USA
Email: bill.qubeck@sycamoreinformatics.com

Website: www.sycamoreinformatics.com