EVOLVING SUBMISSION STANDARDS: TRANSITIONING FROM XPT TO DATASET-JSON

Gomathi S

Statistical Programmer I, ICON PLC

PharmaSUG SDE APRIL 11, 2026

DISCLAIMER

- The Opinions expressed in this presentation and on the following slides are solely those of the presenter and not necessarily those of ICON. ICON does not guarantee the accuracy or reliability of the information provided herein.

AGENDA

- Background
- Limitations of XPT
- Introduction to Dataset-JSON
- Why Dataset-JSON?
- Example of Dataset-JSON
- Metadata Attributes
- How to implement in R
- Adoption and Future scope

BACKGROUND

1989

In 1989 SAS XPORT (XPT) version 5 is established

1999

In 1999, XPT V5 became the standard for FDA submission

Oct. 2012

Version 8: Introduced in October 2012 to address limitations like shorter character fields

Nov. 2012

In November 2012, FDA held a public meeting to look for alternatives to XPT.

BACKGROUND – CONTD.,

2014–2015

In 2014- 2015: considered Dataset-XML as an alternative

2022

In 2022, after assessment FDA identified Dataset-JSON as the promising alternative to XPT V5

2023–2024

FDA began pilot project with CDISC and PHUSE to test dataset-JSON for regulatory submissions

9th April, 2025

FDA published a federal notice formally requesting comments on Dataset-JSON v1.1 as a new exchange standard

LIMITATIONS OF XPT

- Data file format
  - Limited variable types
  - Limited to ASCII encoding
  - 8 character variable names, 40 character labels and 200 character variable width
- Storage
  - Inefficient storage space
  - The inability to compress datasets
- Content
  - Lacks a robust metadata layer

INTRODUCTION TO DATASET-JSON

- A CDISC data exchange standard for sharing tabular dataset using JSON.
- To support a broad range of data exchange scenarios.
- Supports API and file-based data exchange.
- Easy to handle and very convenient to use.

CDISC is pleased to announce the release of Dataset-JSON v1.1. Dataset-JSON is a data exchange standard for sharing tabular data using JSON. It is designed to meet a wide range of data exchange scenarios, including regulatory submissions and API-based data exchange.

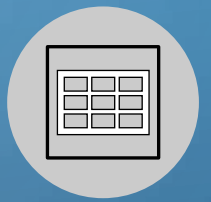
Features of Dataset-JSON v1.1

Each Dataset-JSON dataset can optionally reference a Define-XML document containing more complete metadata for the dataset. Dataset-JSON is based on the JSON standard that is simple to implement, very stable, and widely supported. It is also designed to address the limitations of legacy formats and is extensible to support new metadata and new use cases.

WHY DATASET-JSON?

Easily implemented

Very stable and widely supported in most programming languages.



It works with very minimal dependencies.

Language independent standard.

JSON is easy to read and write

Address SAS XPT limitations.

WHY DATASET-JSON? – CONTD.,

Accommodates datasets generated by EDC, ePRO, laboratories and other data sources.

API will provide the most common means to exchange data

Aligns with standards

DATASET-JSON AND Define.XML

- Dataset-JSON includes metadata.
- Metadata in Dataset-JSON and Define.xml should be same
- We can reference Define.xml in Dataset-JSON
  - “metaDataRef” : “Define.xml”
  - URL also can be mentioned

EXAMPLE OF DATASET-JSON


```
{
  "datasetJSONCreationDateTime": "2026-03-22T22:58:09",
  "datasetJSONVersion": "1.1.0",
  "itemGroupOID": "IG.IRIS",
  "records": 6,
  "name": "IRIS",
  "label": "Iris",
  "columns": [
    {
      "itemOID": "IT.IR.Sepal.Length",
      "name": "Sepal.Length",
      "label": "Sepal Length",
      "dataType": "float",
      "keySequence": 2
    },
    {
      "itemOID": "IT.IR.Sepal.Width",
      "name": "Sepal.Width",
      "label": "Sepal Width",
      "dataType": "float"
    },
    {
      "itemOID": "IT.IR.Petal.Length",
      "name": "Petal.Length",
      "label": "Petal Length",
      "dataType": "float",
      "keySequence": 3
```

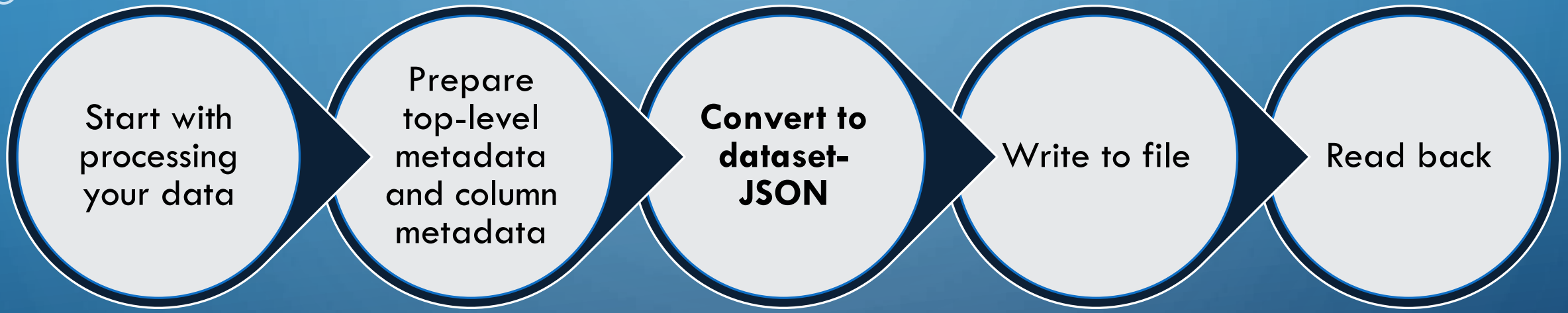

Top-level metadata

Column Metadata


```
    {
      "itemOID": "IT.IR.Species",
      "name": "Species",
      "label": "Flower Species",
      "dataType": "string",
      "length": 10,
      "keySequence": 1
    }
  ],
  "rows": [
    [
      5.1,
      3.5,
      1.4,
      0.2,
      "setosa"
    ],
    [
      4.9,
      3.0,
      1.4,
      0.2,
      "setosa"
    ],
    [
      4.7,
      3.2,
```


Rows

WORKING WITH DATASET-JSON



- In R we have packages like {datasetjson} for this conversion
- SAS and Python also have conversion packages and tools.

INTRODUCTION TO {datasetjson} PACKAGE

- Developed to Read and Write CDISC Dataset-JSON Files by Atorus Research and Johnson & Johnson

Functions	Usage
dataset_json	Creates base object used to write a Dataset-JSON file.
write_dataset_json	Write out a Dataset JSON file
read_dataset_json	Read a Dataset JSON to datasetjson object
set_variable_attributes	Using the columns element of the Dataset-JSON file, assign the available metadata to individual columns
get_column_metadata	This function pulls out the column metadata from the datasetjson object in data.frame.
validate_dataset_json	to retrieve the error information of an invalid JSON file per the Dataset JSON schema

TOP-LEVEL METADATA ATTRIBUTES

Attribute	Description
datasetJSONCreationDateTime	The date/time file was created
datasetJSONVersion	The version of the dataset-JSON standard used to create the dataset.
itemGroupOID	Key value is a unique identifier for Dataset, corresponding to ItemGroupDef/@OID in Define-XML."
records	Number of records present in the dataset
name	Dataset name
label	Dataset label
columns	Variable level metadata for the Dataset JSON object.


```
{
  "datasetJSONCreationDateTime": "2026-04-08T08:55:06",
  "datasetJSONVersion": "1.1.0",
  "itemGroupOID": "IG.IRIS",
  "records": 6,
  "name": "IRIS",
  "label": "Iris",
  "columns": [
    {
```

OPTIONAL TOP-LEVEL METADATA ATTRIBUTES

ATTRIBUTE	DESCRIPTION
fileOID	fileOID parameter, defined as "A unique identifier for this file."
lastModified	The date/time the source database was last modified before creating the DatasetJSON file
originator	originator parameter, defined as "The organization that generated the DatasetJSON file."
study	Study OID value
MetaDataVersion	Metadata version OID value
metaDataRef	Metadata reference (i.e. path to Define.xml)

COLUMN METADATA


```
"columns":[
  {
   "itemOID":"IT.IR.Sepal.Length",
   "name":"Sepal.Length",
   "label":"Sepal Length",
   "dataType":"float",
   "keySequence":2},
  {
    "itemOID":"IT.IR.Sepal.Width",
    "name":"Sepal.Width",
    "label":"Sepal Width",
    "dataType":"float"},
  {
    "itemOID":"IT.IR.Petal.Length",
    "name":"Petal.Length",
    "label":"Petal Length",
    "dataType":"float",
    "keySequence":3},
  {
```


Attribute	Description
itemOID	Unique identifier for the variable that may also function as a foreign key to an ItemDef/@OID in an associated Define-XML file.
name	Variable name
label	Variable label
datatype	data type of the variable
keySequence	Indicates that this item is a key variable in the dataset structure. It also provides an ordering for the keys.

COLUMN METADATA – CONTD.,

Attribute	Description
targetDataType	Indicates the data type into which the receiving system must transform the associated Dataset-JSON variable.
length	Specifies the number of characters allowed for the variable value when it is represented as a text. Its optional for integers.
displayFormat	A SAS display format value used for data visualization of numeric float and date values.

ROW DATA

- Each record is represented as an array of variable values
- Missing values are represented as NULL
- Empty strings are represented as “”


```
"rows":[
  [5.1,3.5,1.4,0.2,"setosa"],
  [4.9,3.0,1.4,0.2,"setosa"],
  [4.7,3.2,1.3,0.2,"setosa"],
  [4.6,3.1,1.5,0.2,"setosa"],
  [5.0,3.6,1.4,0.2,"setosa"],
  [5.4,3.9,1.7,0.4,"setosa"]]}
```


Petal.Width	Species
0.2	setosa
0.2	setosa
0.2	setosa
0.2	setosa
0.2	setosa
NA	


```
[
  5.4,
  3.9,
  1.7,
  null,
  ""
]
```

WORKFLOW IN R USING {datasetjson} PACKAGE

1. Preparing the dataset

```r
#Just selecting first 6 records and few variables
adsl <- head(pharmaverseadam::adsl) %>%
  select(STUDYID, USUBJID, SUBJID, SITEID, COUNTRY)
```

2. Prepare column metadata

```r
# Creating a data.frame that contains the column metadata
items <- data.frame(
  itemOID = c("STUDYID", "USUBJID", "SUBJID", "SITEID", "COUNTRY"),
  name    = c("STUDYID", "USUBJID", "SUBJID", "SITEID", "COUNTRY"),
  label = c("Study Identifier",
            "Unique Subject Identifier",
            "Subject Identifier",
            "Site Identifier",
            "Country"),
  dataType  = c("string", "string", "string", "string", "string")
)
```

WORKFLOW IN R – CONTD.,

3. Creating dataset-JSON object

```r
# Create dataset-JSON object
adsl_json <- dataset_json(
  adsl,
  item_oid    = "IG.ADSL",
  name = "ADSL",
  dataset_label= "Subject-Level Analysis Dataset",
  columns = items
)
```

4. Writing to file path

```r
# Write to file
write_dataset_json(adsl_json,
                   "adsl.json",
                   pretty = TRUE)
```

Dataset-JSON object we created looks like below


```
{
  "datasetJSONCreationDateTime": "2026-03-22T23:09:05",
  "datasetJSONVersion": "1.1.0",
  "itemGroupOID": "IG.ADSL",
  "records": 6,
  "name": "ADSL",
  "label": "Subject-Level Analysis Dataset",
  "columns": [
    {
      "itemOID": "STUDYID",
      "name": "STUDYID",
      "label": "Study Identifier",
      "dataType": "string"
    },
    {
      "itemOID": "USUBJID",
      "name": "USUBJID",
      "label": "Unique Subject Identifier",
      "dataType": "string"
    },
```

WORKFLOW IN R – CONTD.,

5. Reading datasetJSON object

```
# Reading datasetjson object
adsl_json <- read_dataset_json("adsl.json")
```

	STUDYID Study Identifier	USUBJID Unique Subject Identifier	SUBJID Subject Identifier	SITEID Site Identifier	COUNTRY Country
1	CDISCPILOT01	01-701-1015	1015	701	USA
2	CDISCPILOT01	01-701-1023	1023	701	USA
3	CDISCPILOT01	01-701-1028	1028	701	USA
4	CDISCPILOT01	01-701-1033	1033	701	USA
5	CDISCPILOT01	01-701-1034	1034	701	USA
6	CDISCPILOT01	01-701-1047	1047	701	USA

VALIDATING DATA – CONTD.,

- The package has the dataset-JSON schema version 1.1.0 built-in for validation.
- validate_dataset_json(): Validates a Dataset JSON file against the official schema.

```
> adsl_js <- write_dataset_json(adsl_json)
> validate_dataset_json(adsl_js)

File is valid per the Dataset JSON v1.1.0 schema

[1] instancePath schemaPath   keyword      params       message      schema       parentSchema
[8] dataPath
<0 rows> (or 0-length row.names)
```

CHECKING THE DIFFERENCE

```
> diffdf::diffdf(adsl, adsl_json)
Differences found between the objects!

Summary of BASE and COMPARE
  ================================================================================
    PROPERTY              BASE                                  COMP
  --------------------------------------------------------------------------------
      Name                adsl                                adsl_json
     Class      "tbl_df, tbl, data.frame"  "datasetjson_v1_1_0, datasetjso..."
    Rows(#)                 6                                     6
   Columns(#)               5                                     5
  --------------------------------------------------------------------------------


There are columns in BASE and COMPARE with differing attributes !!
  ================================================================================
   VARIABLE  ATTR_NAME             VALUES.BASE                    VALUES.COMP
  --------------------------------------------------------------------------------
    SITEID      label          "Study Site Identifier"          "Site Identifier"
    SUBJID      label     "Subject Identifier for the Stu..."  "Subject Identifier"
  --------------------------------------------------------------------------------


Warning message:
In diffdf::diffdf(adsl, adsl_json) :
There are columns in BASE and COMPARE with differing attributes !!
```

CHECKING THE DIFFERENCE

```
> waldo::compare(adsl, adsl_json, max_diffs = Inf)
`class(old)`: "tbl_df"              "tbl"          "data.frame"
`class(new)`: "datasetjson_v1_1_0" "datasetjson" "data.frame"

`attr(old, '_xportr.df_arg_')` is a character vector ('ADSL')
`attr(new, '_xportr.df_arg_')` is absent

`attr(old, 'label')`: "Subject Level Analysis"
`attr(new, 'label')`: "Subject-Level Analysis Dataset"

`attr(old, 'columns')` is absent
`attr(new, 'columns')` is a list

`attr(old, 'datasetJSONCreationDateTime')` is absent
`attr(new, 'datasetJSONCreationDateTime')` is a character vector ('2026-03-31T13:51:30')

`attr(old, 'datasetJSONVersion')` is absent
`attr(new, 'datasetJSONVersion')` is a character vector ('1.1.0')

`attr(old, 'itemGroupOID')` is absent
`attr(new, 'itemGroupOID')` is a character vector ('IG.ADSL')

`attr(old, 'name')` is absent
`attr(new, 'name')` is a character vector ('ADSL')

`attr(old, 'records')` is absent
`attr(new, 'records')` is an integer vector (6)

`attr(old$SUBJID, 'label')`: "Subject Identifier for the Study"
`attr(new$SUBJID, 'label')`: "Subject Identifier"

`attr(old$SITEID, 'label')`: "Study Site Identifier"
`attr(new$SITEID, 'label')`: "Site Identifier"
```

NDJSON AND COMPRESSED DATASET-JSON

NDJSON:

- **NDJSON – Newline delimited JSON**
- **Variation of JSON designed for bulk transfers - both can have same content**

Compressed dataset-JSON:

- **Compressed dataset-JSON is based on NDJSON format**
  - **Which makes it easier to process larger datasets**
- **DEFLATE algorithm – SDTM (15 times smaller) and ADaM (18 times smaller)**

ADOPTION AND FUTURE SCOPE

- Getting Familiarized with Dataset-JSON using available open-source tools
- Evaluate how dataset-JSON will affect EDC, data management and analysis environment
- Develop or adapt conformance rules for dataset-JSON
- Generate both XPT and dataset-JSON while transition period

REFERENCES

- Converting from XPT • datasetjson
- The Future of Clinical Data Exchange: CDISC Dataset-JSON v1.1 | Clinical Standards Hub
- Breaking Free from the XPT: Exploration of Dataset-JSON as an Alternative Transport File to Regulatory Agencies – Breaking Free from the XPT
- Transport-for-the-Next-Generation-Version-1.0.pdf
- WP-88+Dataset-JSON+Report.pdf
- GitHub - atorus-research/datasetjson workshop: R/Pharma Dataset JSON workshop content · GitHub

THANK YOU

- E-mail: gomathisudhakar1998@gmail.com
- LinkedIn: Gomathi S