Dataset-JSON SAS© implementation

Lex Jansen

About Lex Jansen

- 16 years in an IT/Standards role in Biostatistics at Organon
- 4 years as a consultant to help companies implement CDISC
- 10 years at SAS
  - 7 years as Principal Software Developer working on SAS Clinical Standards Toolkit (implementing mostly XML based standards (Define-XML, ODM, Dataset-XML)) and SAS Life Science Analytics Framework (Java).
  - 3 years as Principal Solution Consultant implementing SAS Life Science Analytics Framework
- Since November 2021: Senior Director, Data Science Development at CDISC (contract through Lex Jansen Consulting LLC)

- Core member of the CDISC Data Exchange Standards team since 2008. Co-lead since November 2021
- Core member of the CDISC Define-XML development team.
  - One of the main Define-XML v2 developers.
  - Developer of CDISC/PhUSE Define-XML v2.x XSL stylesheets.
  - One of the main developers of the Analysis Results Metadata v1.0 for Define-XML v2.0 extension
- Core member of the CDISC Dataset-XML development team.

Agenda

1. Introduction
2. What is JSON
3. Dataset-JSON Document Structure
4. SAS and JSON
5. Writing Dataset-JSON with SAS
6. Reading Dataset-JSON with SAS
7. Conclusion

Introduction

- In **1999**, the FDA standardized the submission of clinical and non-clinical data using the SAS Version 5 XPT Transport Format
- The SAS Version 5 XPT Transport format dates from **1989** and was first available as part of SAS version 5
- Since 1989 there have been many changes to the industry with respect to the process for submissions and the approaches to data curation and manipulation - *but we still submit in SAS Version 5 XPT Transport format*
- For many years it has been recognized that the ASCII-based SAS Version 5 XPT Transport format has some limitations and issues

Introduction - SAS Version 5 XPT Transport files

Technical limitations:

- Current data formats supported are limited to US ASCII (for Character formats) and IBM INTEGER and DOUBLE (for Numeric formats).
- Only supports US ASCII Character encoding. No multibyte characters are supported; this requires translation and/or transcription from the source data
  - Recently SAS changed the specification from: "*All character data are stored in ASCII, regardless of the operating system*" to "*All character data is stored in the Windows encoding that is compatible with the SAS session encoding that is used to create the file*".
  - Although technically character data can be stored in encodings other than ASCII, there is no method of conveying encoding information other than documenting it with the delivery of the transport file.
- Variable names must be alphanumeric, Variable names are limited to a maximum length of 8 characters, Variable labels are restricted to a maximum length of 40 characters
- Character variable data widths are limited to 200 characters

Introduction - SAS Version 5 XPT Transport files

Storage limitations:

- The SAS Version 5 XPT Transport format has a highly inefficient use of storage space. There is often empty space for columns allocated, but not used by data and this can lead to significant wasted space
  - Fixed fields and records that are often padded with blanks
- This inefficiency forces sponsors to re-size character variables to be compliant with regulatory requirements
- The inability to compress datasets leads to significant file logistical issues, due to the requirement that the maximum size of the files is 5 Gigabytes or smaller

```
+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+
HEADER RECORD*******LIBRARY HEADER RECORD!!!!!!!000000000000000000000000000000
SAS     SAS     SASLIB  6.06    bsd4.2                          13APR89:10:20:06
13APR89:10:20:06
HEADER RECORD*******MEMBER  HEADER RECORD!!!!!!!000000000000000001600000000140
HEADER RECORD*******DSCRPTR HEADER RECORD!!!!!!!000000000000000000000000000000
SAS     ABC     SASDATA 6.06    bsd4.2                          13APR89:10:20:06
13APR89:10:20:06
HEADER RECORD*******NAMESTR HEADER RECORD!!!!!!!000000000200000000000000000000
```

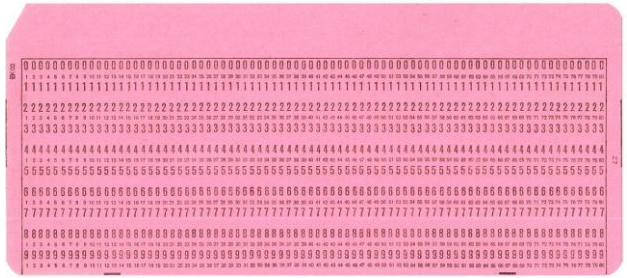


image: Wikimedia Commons

Introduction - SAS Version 5 XPT Transport files

Other limitations:

- The format is only suited for transporting and storing two-dimensional "flat" data structures. This restricts the structure of the content that can be transported.
- The two-dimensional nature of the transport format has led to sub-optimal designs in the structures of the content models (SDTM, ADaM, SEND), that are used to store and transport clinical and nonclinical data
- Even more challenges for other types of data (RWD, …)
- SAS Version 5 Transport format is not an extensible modern technology
- Owned by SAS - not supported and maintained by an open, consensus-based standards development organization

Introduction

- In **2012** FDA organized a Public Meeting on Study Data Exchange
- SAS created the SAS Version 8 XPT Transport format to address some of the issues raised.
- Some of the **SAS Version 5** Transport format limitations have been addressed in the **SAS Version 8** Transport format, e.g. longer character fields, longer names and labels. However, the updated format does not address the other issues and concerns.
- CDISC presented Dataset-XML at the public meeting

Introduction - Dataset-XML

- In April 2014 CDISC published version 1.0 of the Dataset-XML standard
- Dataset-XML is an ODM-based standard format for transporting tabular dataset data in XML between any two entities
- Dataset-XML can be easily extended beyond the tabular two-dimensional format
- FDA and sponsors did a successful pilot with Dataset-XML, although the observation about the Dataset-XML file size seemed an obstacle for the acceptance of this new format.
- Another concern raised about Dataset-XML has been that the metadata - Define-XML - is completely separated from the data. To be able to process a Dataset-XML file one always needs the associated Define-XML document.

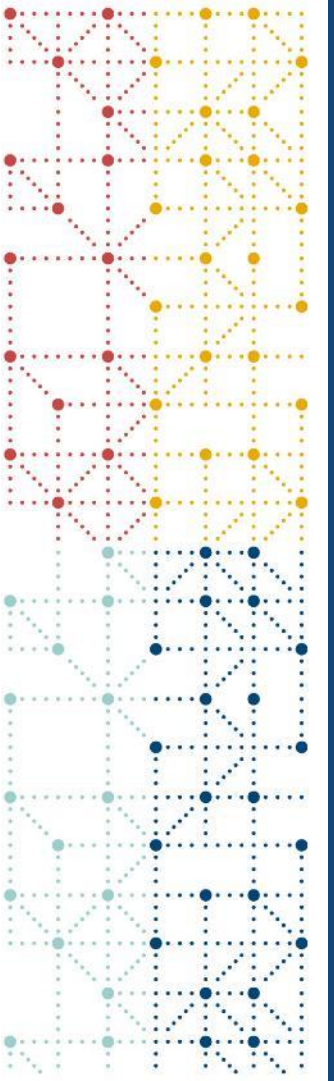
Introduction - Dataset-JSON

- Dataset-JSON was adapted from the Dataset-XML specification but uses JSON format
- Like Dataset-XML, each Dataset-JSON file is connected to the Define-XML document, containing detailed metadata information
- Dataset-JSON files contain basic information about dataset variables, so that it is possible to have a simple view of the contents of a dataset without the need of a Define-XML document
- Dataset-JSON files are also much smaller in size compared to SAS Version 5 XPT Transport files and Dataset-XML files
- JSON is very well positioned to play a role in the digital transition from a file based format (XPT) to an API-based communication protocol
- Most modern web services use JSON as a data exchange format due to the speed and agility it offers

Dataset-JSON vs Dataset-XML vs SAS v5 XPT Size (Kb)

		SAS v5 XPT	Dataset-XML	Dataset-JSON
SDTM	FT	5,917	4,287	858
	LB	2,699	4,104	640
	VS	784	1,372	229
ADaM	ADLBC	33,441	145,575	24,942
	ADQSNPIX	12,840	61,561	8,404
	ADVS	13,313	51,646	8,257

- These are numbers about uncompressed files
- These are exchange formats, not operational formats



What is JSON

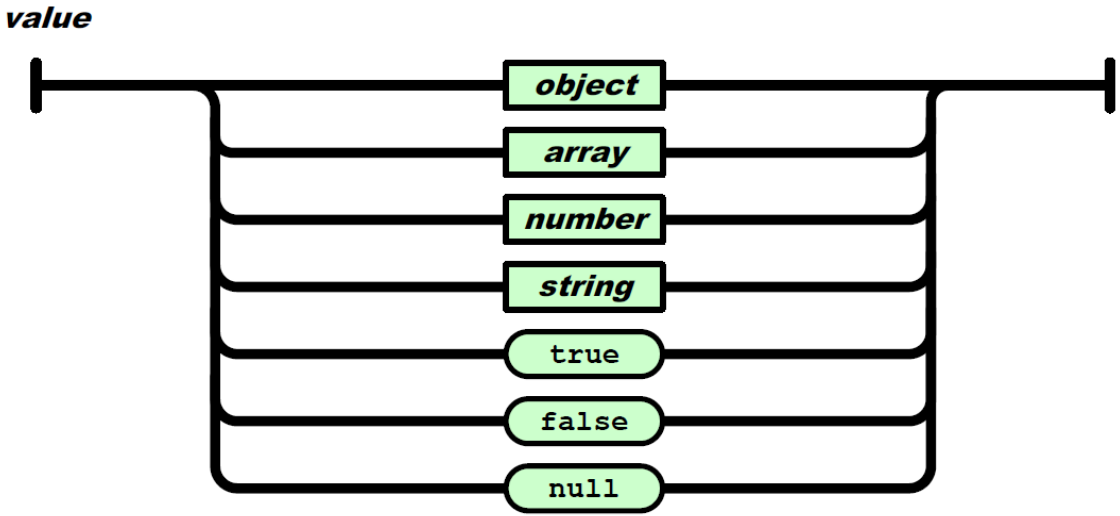
What is JSON

- **J**ava**S**cript **O**bject **N**otation (JSON) is lightweight, text-based, language-independent syntax for defining data interchange formats
- Derived from the JavaScript programming language but is programming language independent
- JSON defines a small set of structuring rules for the portable representation of structured data
- A JSON text is a sequence of tokens formed from **Unicode** code points that conforms to the JSON value grammar
- JSON is a syntax of braces, brackets, colons and commas

- Insignificant whitespace (character tabulation, line feed, carriage return, space) is allowed before or after any token, or within strings, but not within any token

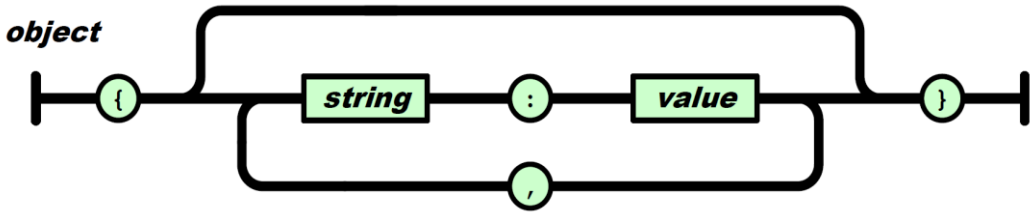
What is JSON

- A JSON value must be an *object*, *array*, *number*, or *string* or one of following literals: `false`, `null`, `true`



A **string** must be wrapped in double quotation marks.

What is JSON - JSON Object




```
{
    "clinicalData": {
        "studyOID": "cdisc.com/CDISCPILOT01",
        "metaDataVersionOID": "MDV.MSGv2.0.SDTMIG.3.3.SDTM.1.7",
        "itemGroupData": {
            "IG.DM": {
                "records": 18,
                "name": "DM",
                "label": "Demographics",
                "items": [ ···
                ]
            }
        }
    }
}
```

What is JSON - JSON Object

- JSON **objects**: simple notation for expressing collections of name/value pairs
- No restrictions on the strings used as names
- No requirement that name strings be unique
- No significance to the ordering of name/value pairs

```
{ name: "STUDYID", label: "Study Identifier" }
```

Invalid "JSON" object – no quotes around "name" and "label"

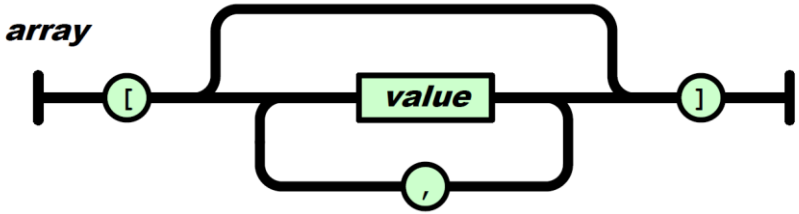
```
{ "name": 'STUDYID', "label": 'Study Identifier' }
```

Invalid "JSON" object – single quotes instead of double quotes

```
{ "name": "STUDYID", "label": "Study Identifier" }
```

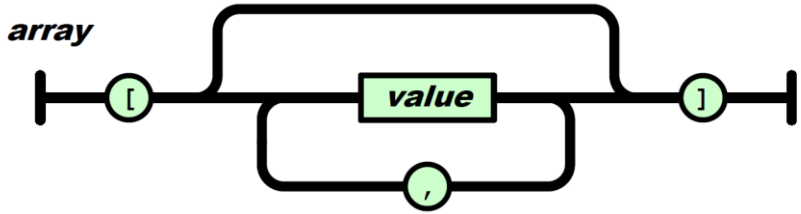
Valid JSON object

What is JSON - JSON Array




```
{
    "clinicalData": {
        "studyOID": "cdisc.com/CDISCPILOT01",
        "metaDataVersionOID": "MDV.MSGv2.0.SDTMIG.3.3.SDTM.1.7",
        "itemGroupData": {
            "IG.DM":
                {"records": 18, "name": "DM", "label": "Demographics",
                 "items": [{"OID": "ITEMGROUPDATASEQ","name": "ITEMGROUPDATASEQ","label": "Record Ide
                 "itemData": [
                     [1,"CDISCPILOT01","DM","CDISC001","1115","2012-11-30",84,"YEARS"],
                     [2,"CDISCPILOT01","DM","CDISC002","1211","2012-11-15",76,"YEARS"],
                     [3,"CDISCPILOT01","DM","CDISC003","1302","2013-08-29",61,"YEARS"],
                     [4,"CDISCPILOT01","DM","CDISC004","1345","2013-10-08",63,"YEARS"]
                 ]
            }
        }
    }
}
```

What is JSON - JSON Array




```
"IG.DM": {
    "records": 18,
    "name": "DM",
    "label": "Demographics",
    "items":
        [
        {"OID": "ITEMGROUPDATASEQ", "name": "ITEMGROUPDATASEQ", "label": "Record Identifier", "type": "integer"},
        {"OID": "IT.DM.STUDYID", "name": "STUDYID", "label": "Study Identifier", "type": "string", "length": 12}
        ]
}
```

What is JSON - JSON Array

- Array structures often used where there is some semantics to the **ordering**
- JSON **arrays** provide support for **ordered** lists of values
- There is no requirement that the values in an array be of the same type.

Invalid "JSON" array – missing values incorrectly represented

```
"itemData": [
    [1, "MyStudy", "001", "DM", , "BLACK"],
    [2, "MyStudy", "002", "DM", 26, ]
]
```

Valid JSON array - for missing values use null or an empty string (string value)

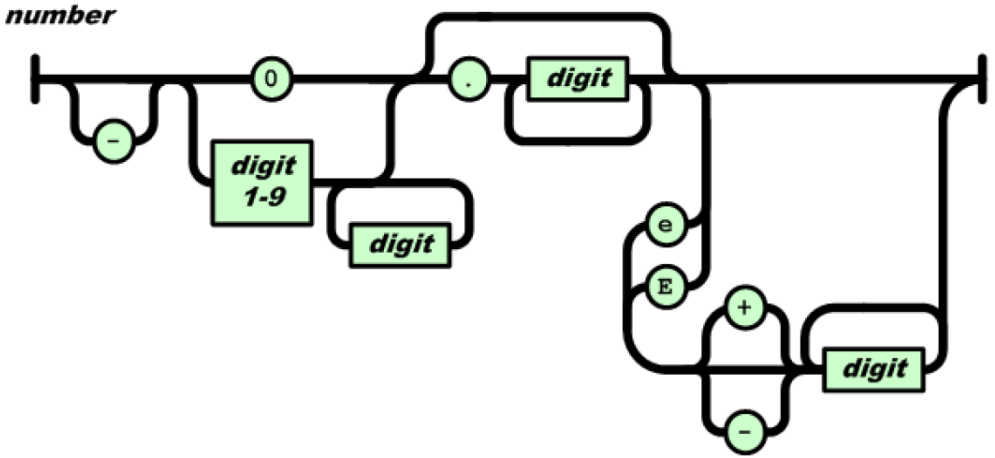
```
"itemData": [
    [1, "MyStudy", "001", "DM", null, "BLACK"],
    [2, "MyStudy", "002", "DM", 26, ""]
]
```

What is JSON - JSON Objects and Arrays

- JSON objects and arrays can be nested
- This means that pretty much any simple or complex data structure can be presented
  - ordinary arrays
  - associative arrays
  - lists
  - symbol tables
  - sets
  - records
  - graphs
  - trees
  - …

What is JSON - Numbers

- **JSON** is agnostic about the semantics of **numbers**.
- A number is a sequence of digits (no double, float, or decimal data types)
- It may have a preceding minus sign, fractional part or exponent



What is JSON - Strings

- A **string** in **JSON is** a sequence of **Unicode** code points wrapped with quotation marks
- Some code points must be escaped by a backslash (\)
  - \" quotation mark
  - \\ backslash
  - \b backspace
  - \f form feed
  - \n line feed
  - \r carriage return
  - \t character tabulation
- Any code point may be represented as a hexadecimal escape sequence
  - The following four cases all produce the same result:

```
"\u002F"  "\u002f"  "\/"  "/"
```

Dataset-JSON Document Structure

Dataset-JSON Document Structure

- Dataset-JSON was adapted from the Dataset-XML specification, but uses JSON format instead of XML
- Specification: https://wiki.cdisc.org/display/ODM2/Dataset-JSON
- GitHub repository with JSON Schema and examples: https://github.com/cdisc-org/DataExchange-DatasetJson

- Each Dataset-JSON file is connected with a Define-XML file, containing **detailed** information about the metadata
- Each Dataset-JSON files contains **basic** information about dataset variables, so that it is possible to have a simple view of the contents of a dataset without the need of a Define-XML document

Dataset-JSON Document Structure

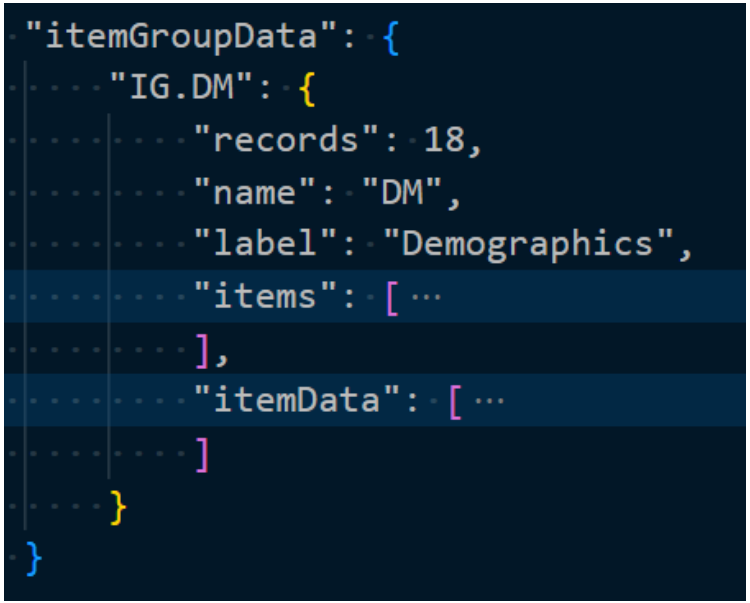
- At the top level of the Dataset-JSON object, there are one of two optional attributes: clinicalData, referenceData.
- Subject data is stored in clinicalData and non-subject data is stored in referenceData.

```
{
    "clinicalData": {
        "studyOID": "cdisc.com/CDISCPILOT01",
        "metaDataVersionOID": "MDV.MSGv2.0.SDTMIG.3.3",
        "itemGroupData": {
            "IG.DM": {…
            }
        }
    }
}
```

```
{
    "referenceData": {
        "studyOID": "cdisc.com/CDISCPILOT01",
        "metaDataVersionOID": "MDV.MSGv2.0.SDTMIG.3.3",
        "itemGroupData": {
            "IG.TS": {…
            }
        }
    }
}
```

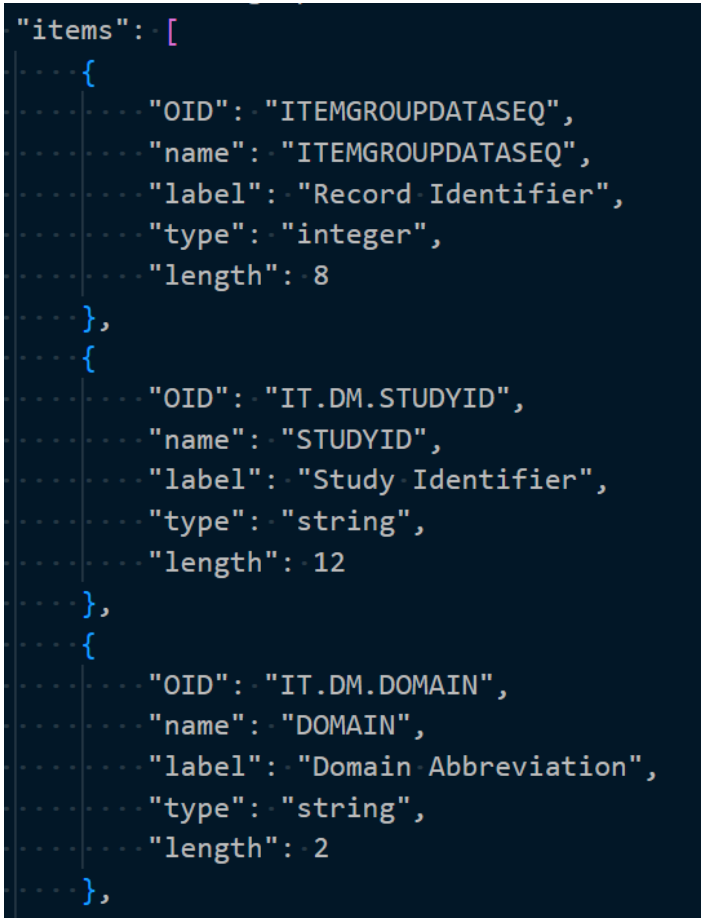
- **studyOID** and **metaDataVersionOID** must match corresponding values in Define-XML

Dataset-JSON Document Structure



- **itemGroupData** is an object with attributes corresponding to individual datasets.
- The attribute name (IG.DM) is the OID of a described dataset, which must be the same as OID of the corresponding itemGroup in the Define-XML file
- **records**, **name**, **label**:
  basic dataset information
- **items** - basic information about variables
- **itemData** - dataset data

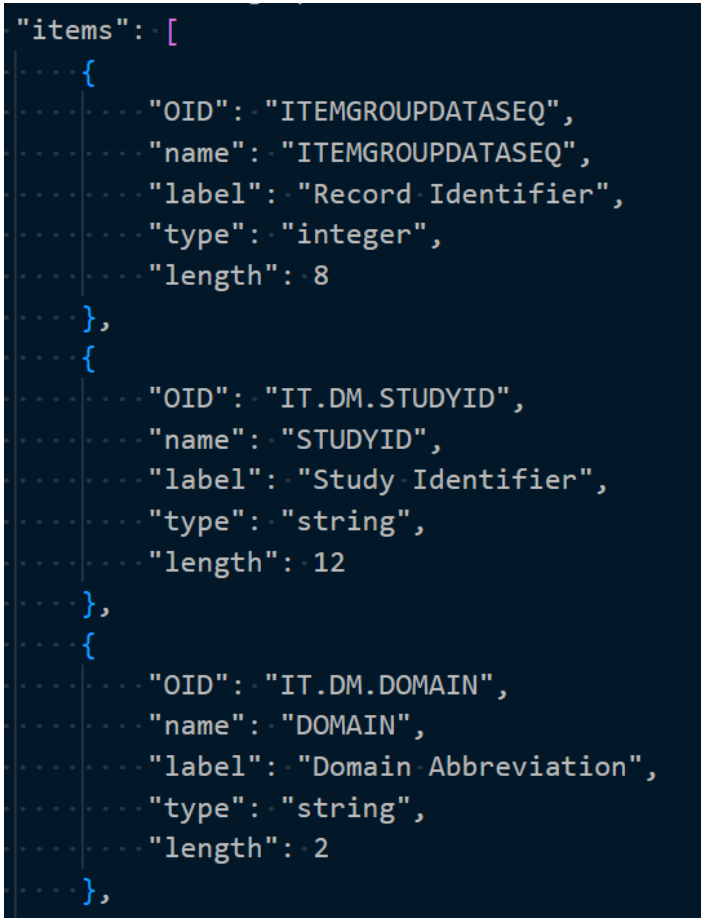
Dataset-JSON Document Structure



```
"items": [
    {
        "OID": "ITEMGROUPDATASEQ",
        "name": "ITEMGROUPDATASEQ",
        "label": "Record Identifier",
        "type": "integer",
        "length": 8
    },
    {
        "OID": "IT.DM.STUDYID",
        "name": "STUDYID",
        "label": "Study Identifier",
        "type": "string",
        "length": 12
    },
    {
        "OID": "IT.DM.DOMAIN",
        "name": "DOMAIN",
        "label": "Domain Abbreviation",
        "type": "string",
        "length": 2
    },
```

- **items** - array of basic information about dataset variables.
- The **order of elements** in the array must be the same as the **order of variables** in the described dataset.
- The first element always describes the Record Identifier (**ITEMGROUPDATASEQ**)

Dataset-JSON Document Structure



```
"items": [
    {
        "OID": "ITEMGROUPDATASEQ",
        "name": "ITEMGROUPDATASEQ",
        "label": "Record Identifier",
        "type": "integer",
        "length": 8
    },
    {
        "OID": "IT.DM.STUDYID",
        "name": "STUDYID",
        "label": "Study Identifier",
        "type": "string",
        "length": 12
    },
    {
        "OID": "IT.DM.DOMAIN",
        "name": "DOMAIN",
        "label": "Domain Abbreviation",
        "type": "string",
        "length": 2
    },
```

- **OID** - OID of a variable (must correspond to the variable OID in the Define-XML file)
- **name** - variable name
- **label** - variable description
- **type** - type of the variable.
  'string', 'integer', 'decimal', 'float', 'double', 'boolean'
- **length** - variable length - most useful for the string type

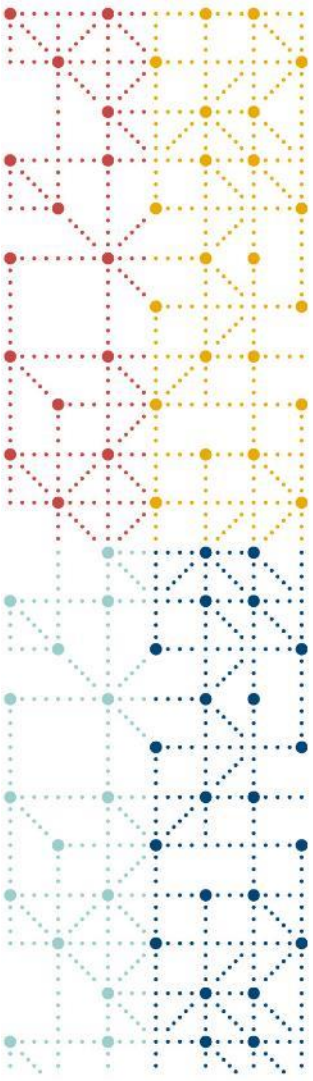
The **length** attribute is optional

Dataset-JSON Document Structure

- **itemData** is an array of records with variables values
- Each record itself is represented as an array of variables values
- The first value is a unique sequence number for each record in the dataset
- Missing values are represented by null in the case of numeric variables, and an empty string in case of character variables:

```
[1, "MyStudy", "", "DM", null]
```

```
"itemData": [
  [1,"CDISCPILOT01","DM","CDISC001","1115","","","701","1928",84,"YEARS","M","WHITE","NOT HIS
  [2,"CDISCPILOT01","DM","CDISC002","1211","2013-01-14","Y","701","1936",76,"YEARS","F","WHIT
  [3,"CDISCPILOT01","DM","CDISC003","1302","","","701","1951",61,"YEARS","M","WHITE","NOT HIS
  ...
]
```



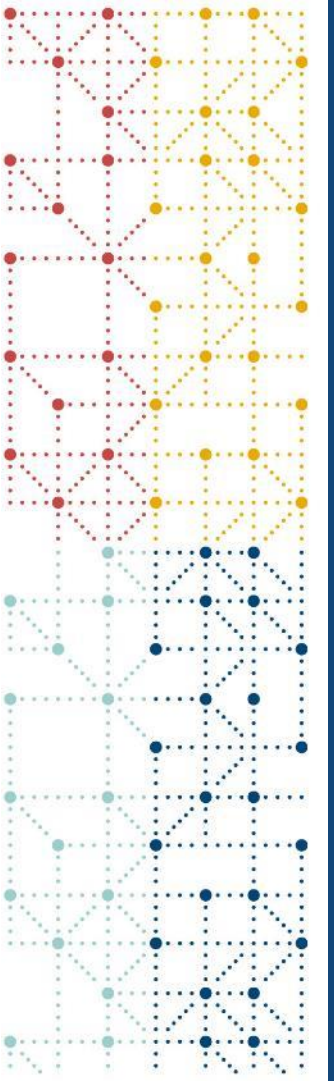
SAS and JSON

SAS and JSON

- In previous papers I showed how to use PROC LUA and a JSON Lua library in SAS to process JSON files:
  - Parsing JSON Files in SAS© Using PROC LUA (PHUSE 2021)
  - Extracting Data Standards Metadata and Controlled Terminology from the CDISC Library using SAS with PROC LUA (PharmaSUG 2021)
- The Lua JSON libraries are an efficient way to manage complex JSON files in SAS that are not too big
- Using SAS can cumbersome when dealing with complex JSON files as it requires the merging of many datasets and dealing with the management of JSON MAP files to correct decisions that the SAS JSON automapper makes in terms of variable types and variable lengths
- PROC LUA reads the entire JSON file into a Lua table that is kept in memory
- This works well with REST APIs and metadata tables but can lead to issues when working with very large datasets

SAS and JSON

- Since the Dataset-JSON format is simple and files might get very large I used native SAS© technology for reading and writing Dataset-JSON files
- It was always possible to create output compliant with the JSON structure using SAS put statements or to parse JSON files in a data step --- *a very tedious process!*
- Starting in SAS© 9.4, you can use **PROC JSON to** write SAS data sets to JSON files
- Starting in SAS© 9.4TS1M4, you can use the **JSON engine** to read JSON files into SAS data sets



Writing Dataset-JSON with SAS

Writing Dataset-JSON with SAS

- PROC JSON in SAS© gives the user control over the JSON output:
  - through the utilization of options
  - through the ability to control containers (objects or arrays)
  - by writing directly to the output file
  - by choosing exactly what to include or not include in the resulting JSON file

```
PROC JSON OUT=fileref | "external-file" <options>;
  EXPORT <libref.>SAS-data-set <(data-set-options)> </options>;
  WRITE VALUES value(s) </options>;
  WRITE OPEN type;
  WRITE CLOSE;
RUN;
```

Writing Dataset-JSON with SAS

PROC JSON options (I used the underlined)

- **PRETTY** | **NOPRETTY** - how to format the JSON output
- **FMTxxx** | **NOFMTxxx** - whether to apply a character, numeric or date/time SAS format to the resulting output if a such a SAS format is associated with a SAS data set variable
- **KEYS** | **NOKEYS** - whether exported observations are written as JSON objects or as JSON arrays
- **SASTAGS** | **NOSASTAGS** - include or suppress SAS metadata
- **SCAN** | **NOSCAN** - scans and encodes input strings to ensure that only characters that are acceptable are exported to the JSON output file
- **TRIMBLANKS** | **NOTRIMBLANKS** - remove or retain trailing blanks from the end of character data in the JSON OUTPUT

Writing Dataset-JSON with SAS

- The **EXPORT** statement
  - identifies the SAS data set to be exported
  - allows to control the resulting output by using options that are specific to PROC JSON
  - Allows to control SAS data set options that are applied to the input SAS data set.
- Same options as on the previous slide (except PRETTY / NOPRETTY)
- If the EXPORT statement is the first statement after the PROC JSON statement, the top-level container is a JSON **object**
- if the NOSASTAGS option is specified in either the PROC JSON or the EXPORT statement, the top-level container is a JSON **array**

Writing Dataset-JSON with SAS

- The **WRITE** statement allows the user to write one or more literal values to the JSON output file
- The value can be either a string, a number, a Boolean value (TRUE or FALSE), or NULL
- SCAN | NOSCAN and TRIMBLANKS | NOTRIMBLANKS can be specified
- The **WRITE OPEN** <type> and **WRITE CLOSE** statements allow the user to open, close, and nest containers
- <type> in the **WRITE OPEN** statement can be either **ARRAY** or **OBJECT**

Writing Dataset-JSON with SAS

```
PROC JSON OUT=jsonfout PRETTY;
  WRITE OPEN OBJECT;
  WRITE VALUES "clinicalData";
  WRITE OPEN OBJECT;
  WRITE VALUES "studyOID" "<study OID>";
  WRITE VALUES "metaDataVersionOID" "<MDV OID>";
  WRITE VALUES "itemGroupData";
  WRITE OPEN OBJECT;
  WRITE VALUES "IG.DM";
  WRITE OPEN OBJECT;
  WRITE VALUES "records" 18;
  WRITE VALUES "name" "DM";
  WRITE VALUES "label" "Demographics";
  WRITE VALUES "items";
  WRITE OPEN ARRAY;
    EXPORT work.column_metadata / KEYS;
  WRITE CLOSE;
  WRITE VALUES "itemData";
  WRITE OPEN ARRAY;
    EXPORT work.column_data / NOKEYS;
  WRITE CLOSE;
  WRITE CLOSE;
  WRITE CLOSE;
  WRITE CLOSE;
  WRITE CLOSE;
RUN;
```

```
{
  "clinicalData": {
    "studyOID": "<study OID>",
    "metaDataVersionOID": "<MDV OID>",
    "itemGroupData": {
      "IG.DM": {
        "records": 18,
        "name": "DM",
        "label": "Demographics",
        "items": [
          column metadata
        ],
        "itemData": [
          Column data
        ]
      }
    }
  }
}
```

Writing Dataset-JSON with SAS

```
PROC JSON OUT=jsonfout PRETTY;
  WRITE OPEN OBJECT;
  WRITE VALUES "clinicalData";
  WRITE OPEN OBJECT;
  WRITE VALUES "studyOID" "<study OID>";
  WRITE VALUES "metaDataVersionOID" "<MDV OID>";
  WRITE VALUES "itemGroupData";
  WRITE OPEN OBJECT;
  WRITE VALUES "IG.DM";
  WRITE OPEN OBJECT;
  WRITE VALUES "records" 18;
  WRITE VALUES "name" "DM";
  WRITE VALUES "label" "Demographics";
  WRITE VALUES "items";
  %* Use macro to avoid creating null values for
     missing attributes;
  %write_json_metadata_array(work.column_metadata);
  WRITE CLOSE;
  WRITE VALUES "itemData";
  WRITE OPEN ARRAY;
    EXPORT work.column_data / NOKEYS;
  WRITE CLOSE;
  WRITE CLOSE;
  WRITE CLOSE;
  WRITE CLOSE;
  WRITE CLOSE;
RUN;
```

```
{
  "clinicalData": {
    "studyOID": "<study OID>",
    "metaDataVersionOID": "<MDV OID>",
    "itemGroupData": {
      "IG.DM": {
        "records": 18,
        "name": "DM",
        "label": "Demographics",
        "items": [
          column metadata
        ],
        "itemData": [
          Column data
        ]
      }
    }
  }
}
```

Writing Dataset-JSON with SAS - metadata

VIEWTABLE: Metadata.Metadata_study

	studyOID	metaDataVersionOID
1	cdisc.com/CDISCPILOT01	MDV.MSGv2.0.SDTMIG.3.3.SDTM.1.7

VIEWTABLE: Metadata.Metadata_tables

	oid	name	label	domain	repeating	isreferencedata	
1	IG.TA	TA	Trial Arms	TA	No	Yes	One record per planned
2	IG.TE	TE	Trial Elements	TE	No	Yes	One record per planned
3	IG.TI	TI	Trial Inclusion/Exclusion Criteria	TI	No	Yes	One record per I/E criter
4	IG.TS	TS	Trial Summary	TS	No	Yes	One record per trial summ value
5	IG.TV	TV	Trial Visits	TV	No	Yes	One record per planned
6	IG.DM	DM	Demographics	DM	No	No	One record per subject
7	IG.SE	SE	Subject Elements	SE	Yes	No	One record per actual El subject
8	IG.SV	SV	Subject Visits	SV	Yes	No	One record per actual vi
9	IG.CM	CM	Concomitant Medications	CM	Yes	No	One record per recorded occurrence or constant- per subject
10	IG.EC	EC	Exposure as Collected	EC	Yes	No	One record per constant

Writing Dataset-JSON with SAS - column metadata

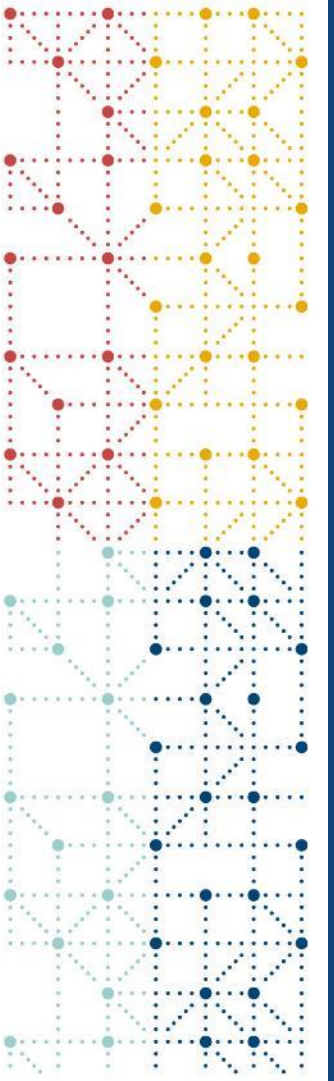
VIEWTABLE: Work.Column_metadata

	OID	name	label	type	length
1	ITEMGROUPDATASEQ	ITEMGROUPDATASEQ	Record Identifier	integer	8
2	IT.VS.STUDYID	STUDYID	Study Identifier	string	12
3	IT.VS.DOMAIN	DOMAIN	Domain Abbreviation	string	2
4	IT.VS.USUBJID	USUBJID	Unique Subject Identifier	string	8
5	IT.VS.VSSEQ	VSSEQ	Sequence Number	integer	3
6	IT.VS.VSTESTCD	VSTESTCD	Vital Signs Test Short Name	string	6
7	IT.VS.VSTEST	VSTEST	Vital Signs Test Name	string	24
8	IT.VS.VSPOS	VSPOS	Vital Signs Position of Subject	string	8
9	IT.VS.VSORRES	VSORRES	Result or Finding in Original Units	string	8
10	IT.VS.VSORRESU	VSORRESU	Original Units	string	9
11	IT.VS.VSSTRESC	VSSTRESC	Character Result/Finding in Std Format	string	200
12	IT.VS.VSSTRESN	VSSTRESN	Numeric Result/Finding in Standard Units	decimal	8
13	IT.VS.VSSTRESU	VSSTRESU	Standard Units	string	9
14	IT.VS.VSSTAT	VSSTAT	Completion Status	string	8
15	IT.VS.VSLOC	VSLOC	Location of Vital Signs Measurement	string	11
16	IT.VS.VSLOBXFL	VSLOBXFL	Last Observation Before Exposure Flag	string	1
17	IT.VS.VSREPNUM	VSREPNUM	Repetition Number	integer	8
18	IT.VS.VISITNUM	VISITNUM	Visit Number	decimal	8
19	IT.VS.VISIT	VISIT	Visit Name	string	200
20	IT.VS.EPOCH	EPOCH	Epoch	string	9
21	IT.VS.VSDTC	VSDTC	Date/Time of Measurements	string	.
22	IT.VS.VSDY	VSDY	Study Day of Vital Signs	integer	8

Writing Dataset-JSON with SAS - column data

VIEWTABLE: Work.Column_data

	ITEMGROUPDATASEQ	STUDYID	DOMAIN	USUBJID	VSSEQ	VSTESTCD	VSTEST	VSPOS	VSORRES
1	1	CDISCPILOT01	VS	CDISC001	1	DIABP	Diastolic Blood Pressure	STANDING	71
2	2	CDISCPILOT01	VS	CDISC001	2	DIABP	Diastolic Blood Pressure	STANDING	71
3	3	CDISCPILOT01	VS	CDISC001	3	DIABP	Diastolic Blood Pressure	STANDING	83
4	4	CDISCPILOT01	VS	CDISC001	4	DIABP	Diastolic Blood Pressure	STANDING	79
5	5	CDISCPILOT01	VS	CDISC001	5	DIABP	Diastolic Blood Pressure	STANDING	68
6	6	CDISCPILOT01	VS	CDISC001	6	DIABP	Diastolic Blood Pressure	SUPINE	77
7	7	CDISCPILOT01	VS	CDISC001	7	DIABP	Diastolic Blood Pressure	STANDING	71
8	8	CDISCPILOT01	VS	CDISC001	8	DIABP	Diastolic Blood Pressure	STANDING	69
9	9	CDISCPILOT01	VS	CDISC001	9	DIABP	Diastolic Blood Pressure	SUPINE	76
10	10	CDISCPILOT01	VS	CDISC001	10	DIABP	Diastolic Blood Pressure	STANDING	52
11	11	CDISCPILOT01	VS	CDISC001	11	DIABP	Diastolic Blood Pressure	STANDING	60
12	12	CDISCPILOT01	VS	CDISC001	12	DIABP	Diastolic Blood Pressure	SUPINE	68
13	13	CDISCPILOT01	VS	CDISC001	13	DIABP	Diastolic Blood Pressure	STANDING	72
14	14	CDISCPILOT01	VS	CDISC001	14	DIABP	Diastolic Blood Pressure	STANDING	68
15	15	CDISCPILOT01	VS	CDISC001	15	HEIGHT	Height		71.5
16	16	CDISCPILOT01	VS	CDISC001	16	PULSE	Pulse Rate	STANDING	51
17	17	CDISCPILOT01	VS	CDISC001	17	PULSE	Pulse Rate	STANDING	50
18	18	CDISCPILOT01	VS	CDISC001	18	PULSE	Pulse Rate	STANDING	50



Reading Dataset-JSON with SAS

Reading Dataset-JSON with SAS

- Starting in SAS© 9.4TS1M4, you can use the **JSON engine** to read JSON files into SAS data sets
- A **JSON map** is a file that the JSON engine uses to define the data set structures that are created when reading JSON
- Without a specified map, the **JSON engine** will automatically create a map
  - **AUTOMAP = REUSE** uses the MAP fileref if the file exists. Otherwise, generates a map
  - **AUTOMAP = CREATE** generates a JSON map and writes it to the MAP fileref
- A data set named ALLDATA is created by default when the JSON engine runs. The ALLDATA data set contains all of the information in the JSON file. This can be turned off with the **NOALLDATA** option
- The generated data sets contain ordinal variables, which are key variables that provide a relationship between the data sets to help with merging. Ordinal variables will not be created when **ORDINALCOUNT= NONE**

Reading Dataset-JSON with SAS

```
FILENAME jsonfile <path to the JSON file>";
FILENAME mapfile "<path to the MAP file to be created>";
LIBNAME data_out "<path to the output folder>";

LIBNAME jsonfile JSON FILEREF=jsonfile MAP=mapfile
        AUTOMAP=CREATE NOALLDATA ORDINALCOUNT=NONE;
PROC COPY IN=jsonfile OUT=data_out;
RUN;
```

Reading Dataset-JSON with SAS

```
{
"clinicalData": {
  "studyOID": "<study OID>",
  "metaDataVersionOID": "<MDV OID>",
  "itemGroupData": {
    "IG.DM": {
      "records": 18,
      "name": "DM",
      "label": "Demographics",
      "items": [
```

column metadata

```
      ]
      "itemData": [
```

column data

```
      ]
    }
  }
}
}
```

Creates 4 data sets:

- **CLINICALDATA**
  - studyOID, metaDataVersionOID
- **ITEMGROUPDATA_IG_DM**
  - records, name, label
- **IG_DM_ITEMS**
  - contains column metadata
- **IG_DM_ITEMDATA**
  - contains column data

Reading Dataset-JSON with SAS

ITEMGROUPDATA_IG_DM

	records	name	label
1	18	DM	Demographics

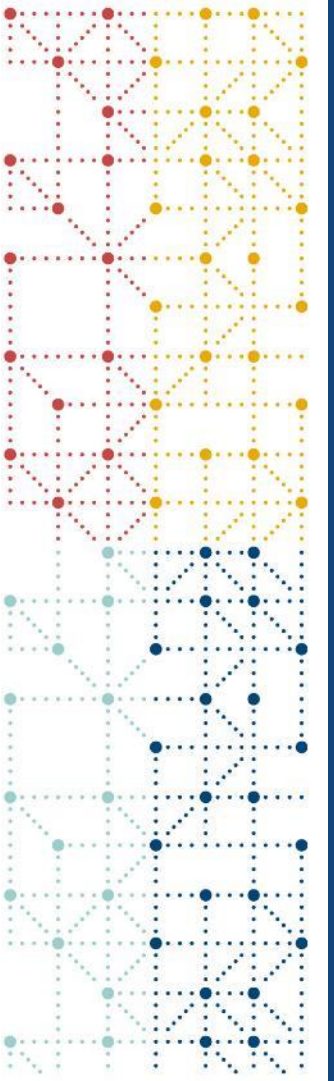
IG_DM_ITEMS

	OID	name	label	type	length
1	ITEMGROUPDATASEQ	ITEMGROUPDATASEQ	Record Identifier	integer	
2	IT.DM.STUDYID	STUDYID	Study Identifier	string	12
3	IT.DM.DOMAIN	DOMAIN	Domain Abbreviation	string	2
4	IT.DM.USUBJID	USUBJID	Unique Subject Identifier	string	8
5	IT.DM.SUBJID	SUBJID	Subject Identifier for the Study	string	4
6	IT.DM.RFSTDTC	RFSTDTC	Subject Reference Start Dat...	string	
7	IT.DM.RFENDTC	RFENDTC	Subject Reference End Dat...	string	
8	IT.DM.RFXSTDTC	RFXSTDTC	Date/Time of First Study Tre...	string	
9	IT.DM.RFXENDTC	RFXENDTC	Date/Time of Last Study Tr...	string	

IG_DM_ITEMDATA

	element1	element2	element3	element4	element5	element6	element7	element8
1	1	CDISCPILOT01	DM	CDISC001	1115	2012-11-30	2013-01-23	2012-11-30
2	2	CDISCPILOT01	DM	CDISC002	1211	2012-11-15	2013-01-14	2012-11-15
3	3	CDISCPILOT01	DM	CDISC003	1302	2013-08-29	2013-11-05	2013-08-29
4	4	CDISCPILOT01	DM	CDISC004	1345	2013-10-08	2014-03-18	2013-10-08
5	5	CDISCPILOT01	DM	CDISC005	1383	2013-02-04	2013-08-06	2013-02-04
6	6	CDISCPILOT01	DM	CDISC006	1429	2013-03-19	2013-04-30	2013-03-19
7	7	CDISCPILOT01	DM	CDISC007	1444	2013-01-05	2013-02-13	2013-01-05
8	8	CDISCPILOT01	DM	CDISC008	1445	2014-05-11	2014-11-01	2014-05-11
9	9	CDISCPILOT01	DM	CDISC009	1087	2012-10-22	2013-04-28	2012-10-22

cdisc



Conclusion

Conclusion

- Dataset-JSON provides an efficient alternative to SAS Version 5 Transport files with full support for UTF-8
- Dataset-JSON is good step towards a modern API-based communication protocol
- SAS fully supports reading and writing Dataset-JSON files

GitHub Repository:
https://github.com/lexjansen/dataset-json-sas

Email:lexjansen@gmail.com
ljansen@cdisc.org

Web: https:\\www.lexjansen.com

LinkedIn: https://www.linkedin.com/in/lexjansen/