Parsing JSON Files in SAS© using PROC LUA

Lex Jansen
SAS Institute Inc., USA

PHUSE EU Connect 2021
Paper AD06

Agenda

- Introduction
- What is JSON
- Reading JSON in SAS
- PROC LUA in SAS
- JSON Lua libraries
- Testing the JSON Lua libraries
- More examples
- Conclusion

Introduction

- JSON is becoming the de-facto standard for data exchanges on the internet, especially in supporting REST APIs
- SAS© can import JSON files into SAS data sets using the JSON LIBNAME engine
- When JSON structures get more complex the use of the JSON LIBNAME engine can result in a large number of SAS data sets that need to be merged.
- This presentation shows an alternative approach for importing JSON structures by using PROC LUA in SAS © with publicly available JSON modules.

What is JSON

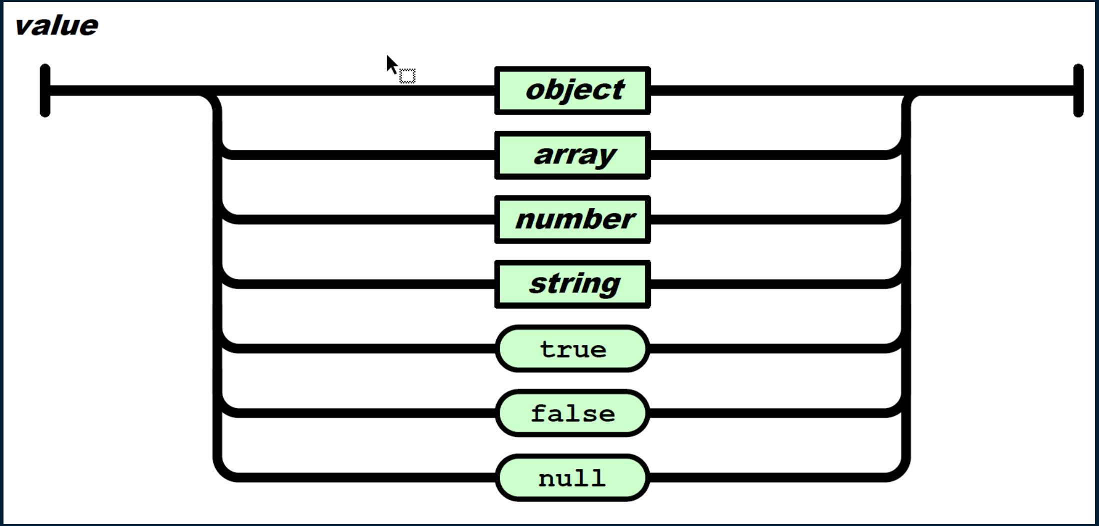
- JavaScript Object Notation (JSON*) is lightweight, text-based, language-independent syntax for defining data interchange formats
- Douglas Crockford originally specified the JSON format in the early 2000s.
- Derived from the JavaScript programming language, but is programming language independent.
- JSON defines a small set of structuring rules for the portable representation of structured data.
- JSON is a syntax of **braces**, **brackets**, **colons** and **commas**

{ } [ ] : ,

*Pronounced /ˈdʒeɪ·sən/, as in “Jason and The Argonauts”.

What is JSON

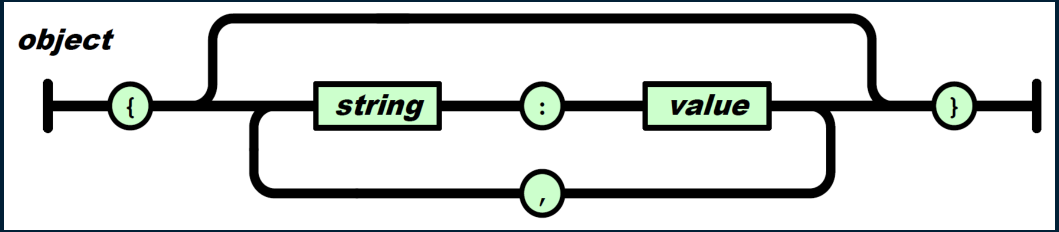
- A JSON value must be an *object*, *array*, *number*, or *string* or one of following literals: `false`, `null`, `true`



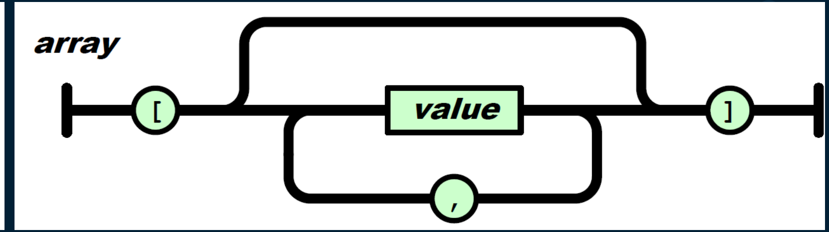
A string must be wrapped in <u>double</u> quotation marks.

What is JSON

object

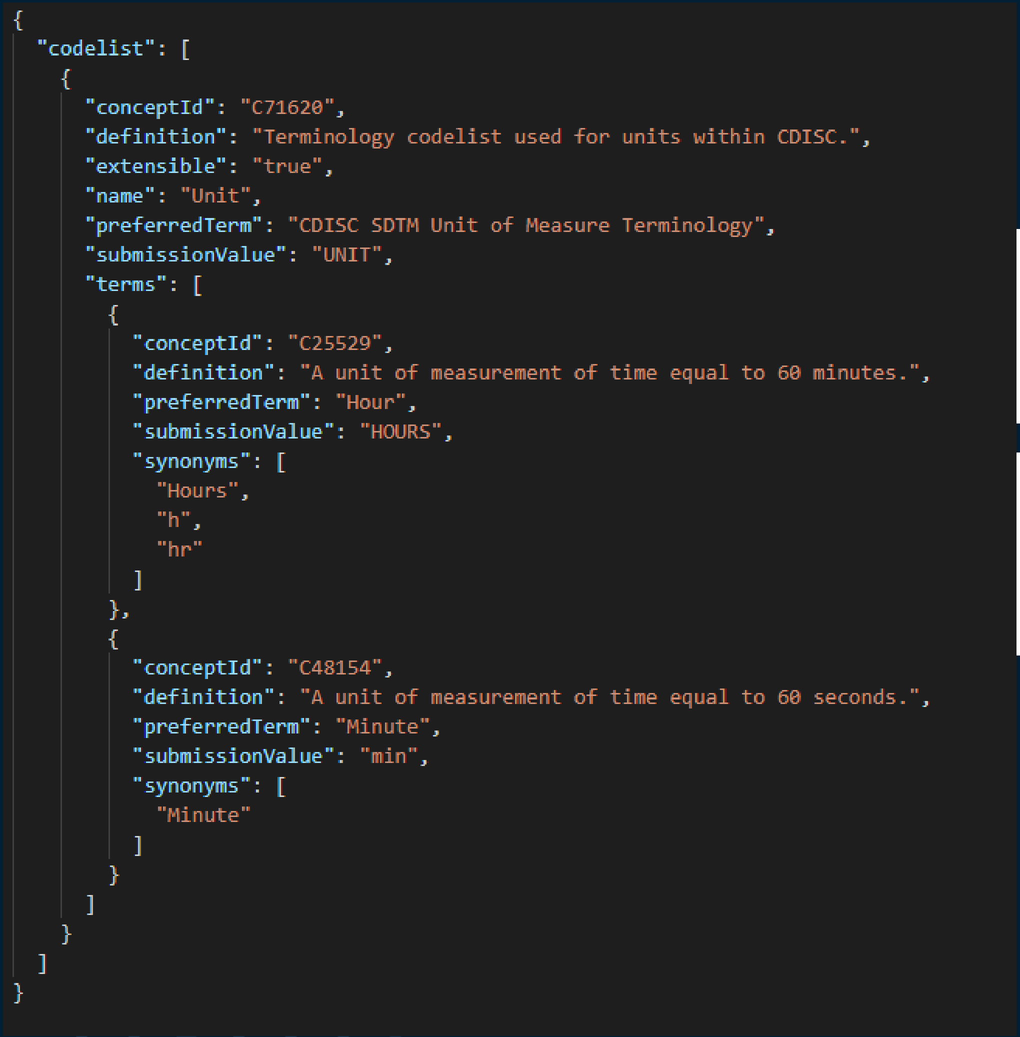


Array

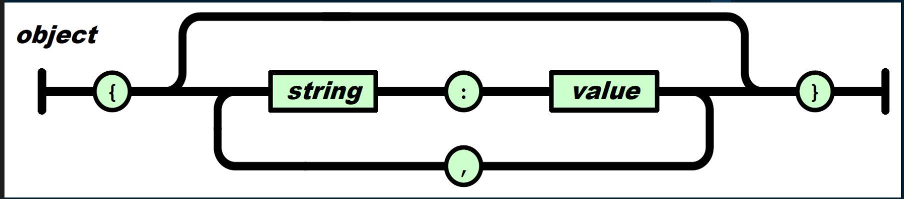


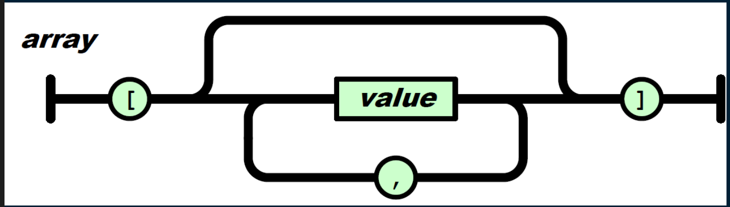
```
{
  "conceptId": "C69112",
  "definition": "A unit of proportion equal to 1E-3. (NCI)",
  "preferredTerm": "Part per Thousand",
  "submissionValue": "ppth",
  "synonyms": [
    "Part per Thousand",
    "per mil",
    "per mille",
    "permil"
  ]
}
```

What is JSON



```
{
  "codelist": [
    {
      "conceptId": "C71620",
      "definition": "Terminology codelist used for units within CDISC.",
      "extensible": "true",
      "name": "Unit",
      "preferredTerm": "CDISC SDTM Unit of Measure Terminology",
      "submissionValue": "UNIT",
      "terms": [
        {
          "conceptId": "C25529",
          "definition": "A unit of measurement of time equal to 60 minutes.",
          "preferredTerm": "Hour",
          "submissionValue": "HOURS",
          "synonyms": [
            "Hours",
            "h",
            "hr"
          ]
        },
        {
          "conceptId": "C48154",
          "definition": "A unit of measurement of time equal to 60 seconds.",
          "preferredTerm": "Minute",
          "submissionValue": "min",
          "synonyms": [
            "Minute"
          ]
        }
      ]
    }
  ]
}
```

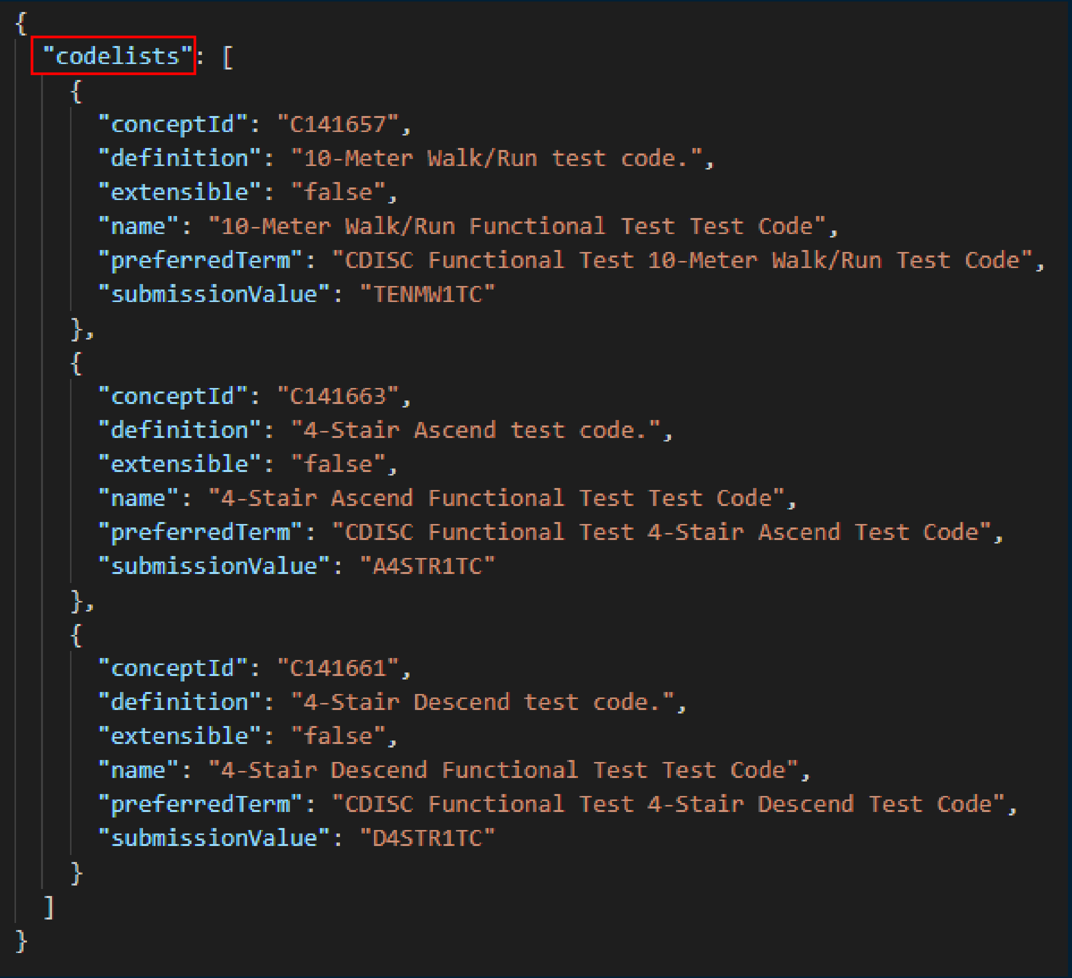




Reading JSON in SAS

- SAS can read JSON files with the JSON LIBNAME engine since SAS 9.4 M4
- Depending on whether the JSON file has a simple or more complex hierarchy, you may want to use a MAP.
- SAS can create a default MAP that you can manually update, if needed
- When JSON files have a more complex hierarchy a large number of data sets may be generated that may need further processing, such as merging, transposing or data step coding.

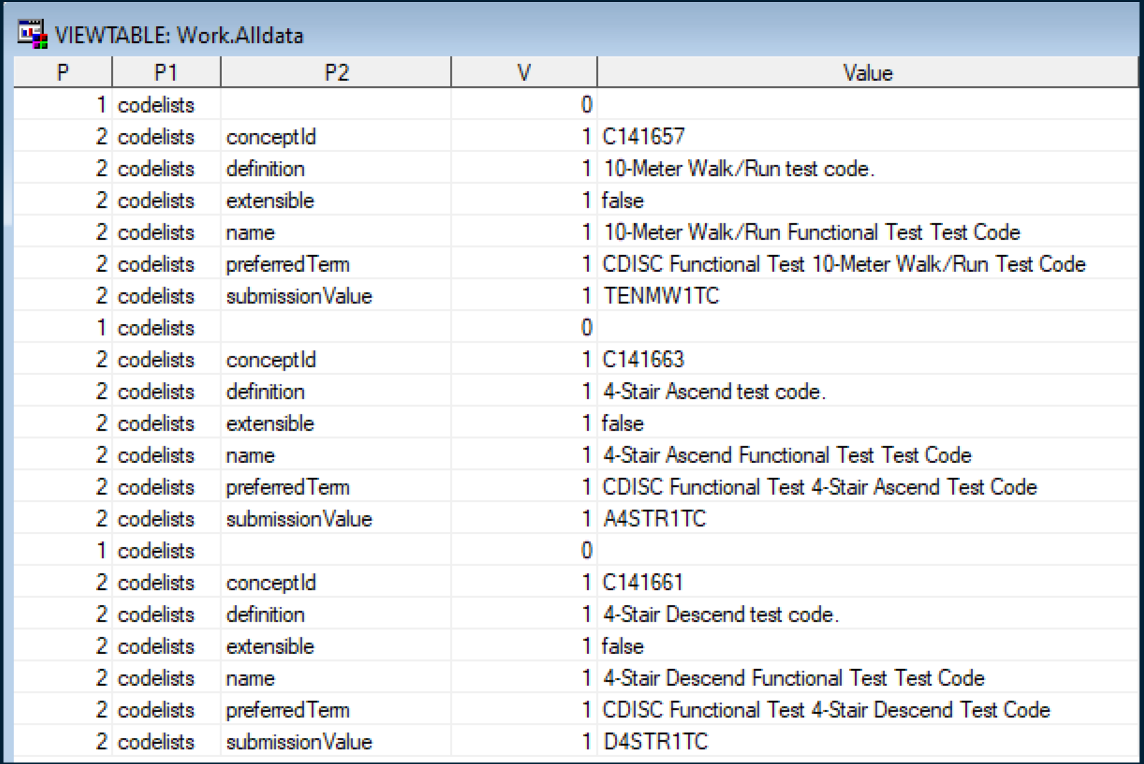
JSON in SAS



```
{
  "codelists": [
    {
      "conceptId": "C141657",
      "definition": "10-Meter Walk/Run test code.",
      "extensible": "false",
      "name": "10-Meter Walk/Run Functional Test Test Code",
      "preferredTerm": "CDISC Functional Test 10-Meter Walk/Run Test Code",
      "submissionValue": "TENMW1TC"
    },
    {
      "conceptId": "C141663",
      "definition": "4-Stair Ascend test code.",
      "extensible": "false",
      "name": "4-Stair Ascend Functional Test Test Code",
      "preferredTerm": "CDISC Functional Test 4-Stair Ascend Test Code",
      "submissionValue": "A4STR1TC"
    },
    {
      "conceptId": "C141661",
      "definition": "4-Stair Descend test code.",
      "extensible": "false",
      "name": "4-Stair Descend Functional Test Test Code",
      "preferredTerm": "CDISC Functional Test 4-Stair Descend Test Code",
      "submissionValue": "D4STR1TC"
    }
  ]
}
```

```
filename jsonfile "&root/example0.json";
libname jsonfile json fileref=jsonfile;

proc copy in=jsonfile out=work;
run;
```

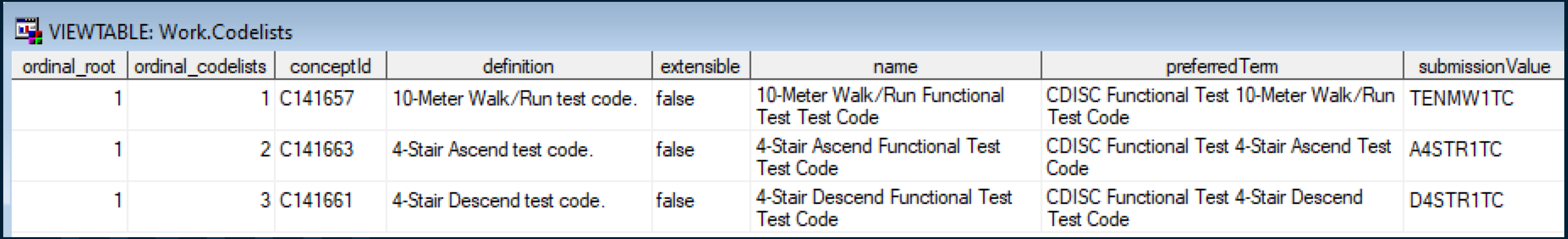
JSON in SAS



VIEWTABLE: Work.Alldata

P	P1	P2	V	Value
1	codelists		0	
2	codelists	conceptId	1	C141657
2	codelists	definition	1	10-Meter Walk/Run test code.
2	codelists	extensible	1	false
2	codelists	name	1	10-Meter Walk/Run Functional Test Test Code
2	codelists	preferredTerm	1	CDISC Functional Test 10-Meter Walk/Run Test Code
2	codelists	submissionValue	1	TENMW1TC
1	codelists		0	
2	codelists	conceptId	1	C141663
2	codelists	definition	1	4-Stair Ascend test code.
2	codelists	extensible	1	false
2	codelists	name	1	4-Stair Ascend Functional Test Test Code
2	codelists	preferredTerm	1	CDISC Functional Test 4-Stair Ascend Test Code
2	codelists	submissionValue	1	A4STR1TC
1	codelists		0	
2	codelists	conceptId	1	C141661
2	codelists	definition	1	4-Stair Descend test code.
2	codelists	extensible	1	false
2	codelists	name	1	4-Stair Descend Functional Test Test Code
2	codelists	preferredTerm	1	CDISC Functional Test 4-Stair Descend Test Code
2	codelists	submissionValue	1	D4STR1TC

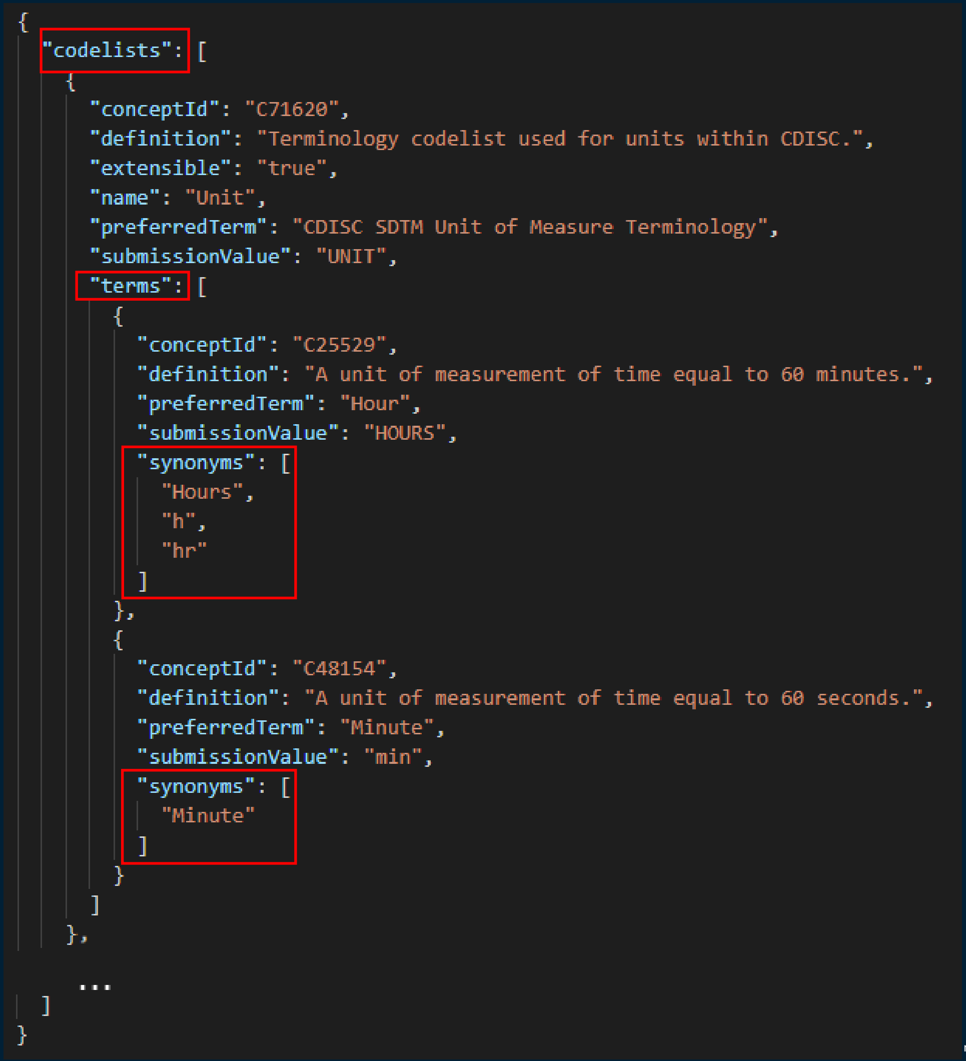
You will get a dataset (Alldata) with all data.



VIEWTABLE: Work.Codelists

ordinal_root	ordinal_codelists	conceptId	definition	extensible	name	preferredTerm	submissionValue
1	1	C141657	10-Meter Walk/Run test code.	false	10-Meter Walk/Run Functional Test Test Code	CDISC Functional Test 10-Meter Walk/Run Test Code	TENMW1TC
1	2	C141663	4-Stair Ascend test code.	false	4-Stair Ascend Functional Test Test Code	CDISC Functional Test 4-Stair Ascend Test Code	A4STR1TC
1	3	C141661	4-Stair Descend test code.	false	4-Stair Descend Functional Test Test Code	CDISC Functional Test 4-Stair Descend Test Code	D4STR1TC

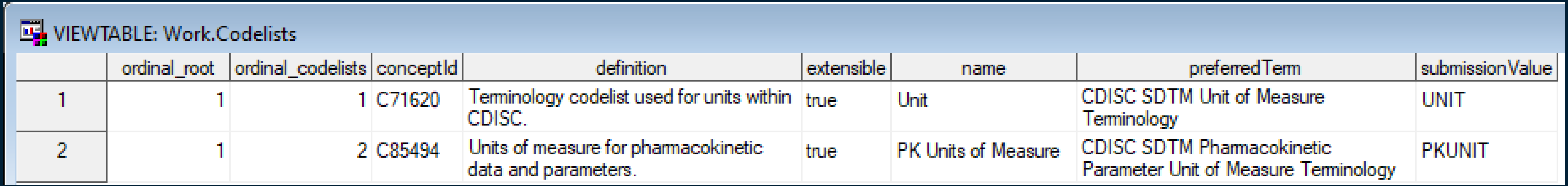
JSON in SAS



```
{
  "codelists": [
    {
      "conceptId": "C71620",
      "definition": "Terminology codelist used for units within CDISC.",
      "extensible": "true",
      "name": "Unit",
      "preferredTerm": "CDISC SDTM Unit of Measure Terminology",
      "submissionValue": "UNIT",
      "terms": [
        {
          "conceptId": "C25529",
          "definition": "A unit of measurement of time equal to 60 minutes.",
          "preferredTerm": "Hour",
          "submissionValue": "HOURS",
          "synonyms": [
            "Hours",
            "h",
            "hr"
          ]
        },
        {
          "conceptId": "C48154",
          "definition": "A unit of measurement of time equal to 60 seconds.",
          "preferredTerm": "Minute",
          "submissionValue": "min",
          "synonyms": [
            "Minute"
          ]
        }
      ]
    },

      ...
  ]
}
```

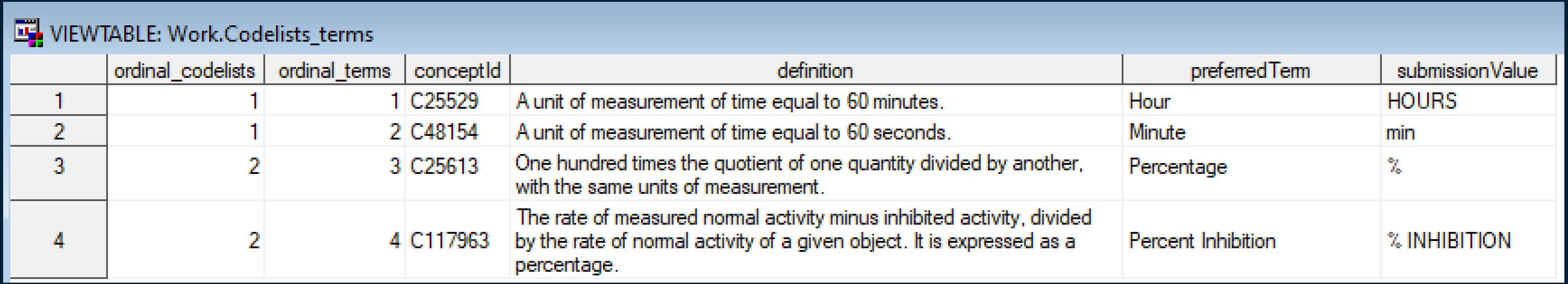
```
filename jsonfile "&root/example1.json";
libname jsonfile json fileref=jsonfile NOALLDATA;

proc copy in=jsonfile out=work;
run;
```

Reading JSON in SAS



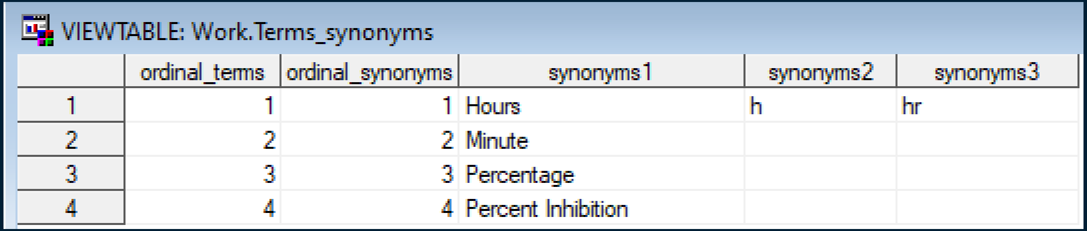
VIEWTABLE: Work.Codelists

	ordinal_root	ordinal_codelists	conceptId	definition	extensible	name	preferredTerm	submissionValue
1	1	1	C71620	Terminology codelist used for units within CDISC.	true	Unit	CDISC SDTM Unit of Measure Terminology	UNIT
2	1	2	C85494	Units of measure for pharmacokinetic data and parameters.	true	PK Units of Measure	CDISC SDTM Pharmacokinetic Parameter Unit of Measure Terminology	PKUNIT



VIEWTABLE: Work.Codelists_terms

	ordinal_codelists	ordinal_terms	conceptId	definition	preferredTerm	submissionValue
1	1	1	C25529	A unit of measurement of time equal to 60 minutes.	Hour	HOURS
2	1	2	C48154	A unit of measurement of time equal to 60 seconds.	Minute	min
3	2	3	C25613	One hundred times the quotient of one quantity divided by another, with the same units of measurement.	Percentage	%
4	2	4	C117963	The rate of measured normal activity minus inhibited activity, divided by the rate of normal activity of a given object. It is expressed as a percentage.	Percent Inhibition	% INHIBITION



VIEWTABLE: Work.Terms_synonyms

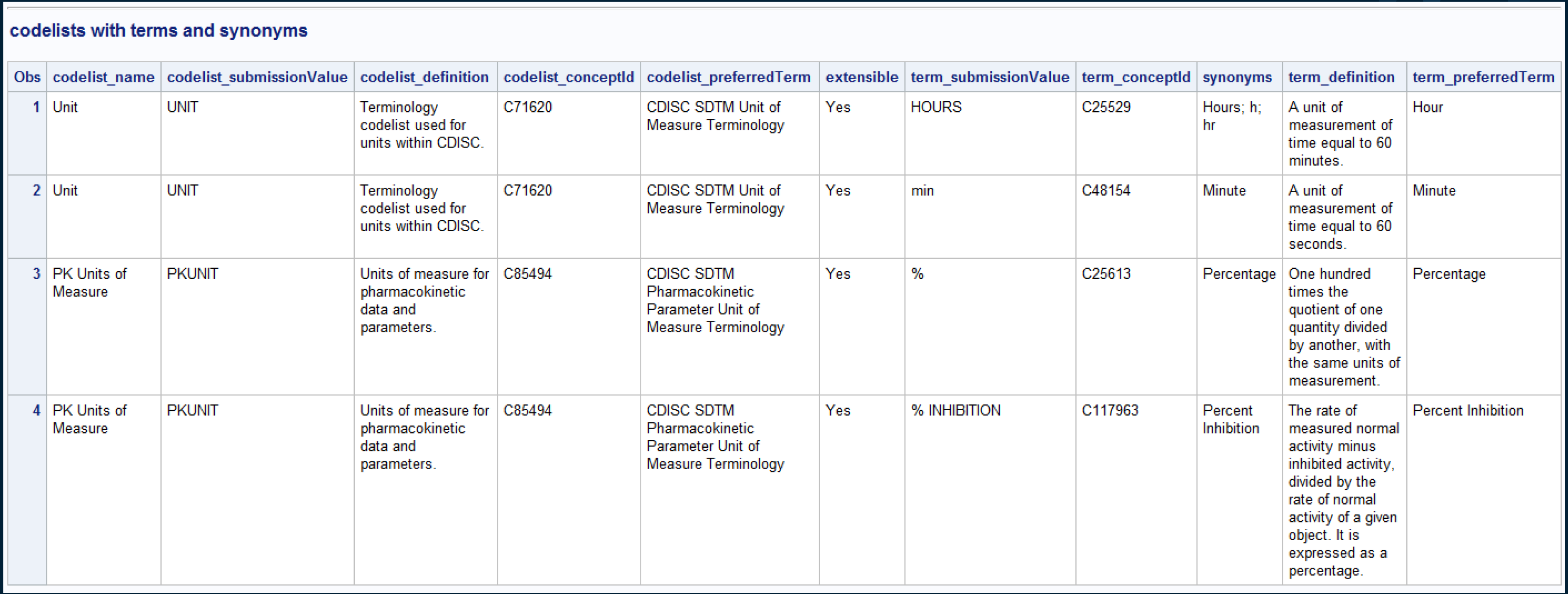
	ordinal_terms	ordinal_synonyms	synonyms1	synonyms2	synonyms3
1	1	1	Hours	h	hr
2	2	2	Minute		
3	3	3	Percentage		
4	4	4	Percent Inhibition		

Reading JSON in SAS

```
proc sql;
  /* concatenate all synonyms* variable names*/
  select cats("s.", name) into :synonym_variables separated by ","
    from dictionary.columns
    where libname = "WORK" and memname eq "TERMS_SYNONYMS" and
      index(upcase(name), "SYNONYMS") and type eq "char";

  create table work.codelist_terms_synonyms
    as select
      c.name as codelist_name,
      c.submissionValue as codelist_submissionValue,
      c.definition as codelist_definition,
      c.conceptId as codelist_conceptId,
      c.preferredTerm as codelist_preferredTerm,
      ifc(c.extensible = "true", "Yes", "No", "") as extensible,
      t.submissionValue as term_submissionValue,
      t.conceptId as term_conceptId,
      catx('; ', &synonym_variables) as synonyms,
      t.definition as term_definition,
      t.preferredTerm as term_preferredTerm
    from work.codelists c
  left join work.codelists_terms t
    on t.ordinal_codelists = c.ordinal_codelists
  left join work.terms_synonyms s
    on s.ordinal_terms = t.ordinal_terms;
quit;
```

Merging the
3 data sets.

JSON in SAS



codelists with terms and synonyms

Obs	codelist_name	codelist_submissionValue	codelist_definition	codelist_conceptId	codelist_preferredTerm	extensible	term_submissionValue	term_conceptId	synonyms	term_definition	term_preferredTerm
1	Unit	UNIT	Terminology codelist used for units within CDISC.	C71620	CDISC SDTM Unit of Measure Terminology	Yes	HOURS	C25529	Hours; h; hr	A unit of measurement of time equal to 60 minutes.	Hour
2	Unit	UNIT	Terminology codelist used for units within CDISC.	C71620	CDISC SDTM Unit of Measure Terminology	Yes	min	C48154	Minute	A unit of measurement of time equal to 60 seconds.	Minute
3	PK Units of Measure	PKUNIT	Units of measure for pharmacokinetic data and parameters.	C85494	CDISC SDTM Pharmacokinetic Parameter Unit of Measure Terminology	Yes	%	C25613	Percentage	One hundred times the quotient of one quantity divided by another, with the same units of measurement.	Percentage
4	PK Units of Measure	PKUNIT	Units of measure for pharmacokinetic data and parameters.	C85494	CDISC SDTM Pharmacokinetic Parameter Unit of Measure Terminology	Yes	% INHIBITION	C117963	Percent Inhibition	The rate of measured normal activity minus inhibited activity, divided by the rate of normal activity of a given object. It is expressed as a percentage.	Percent Inhibition

```
filename map "&root/example1_map.json";
filename jsonfile "&root/example1.json";
libname jsonfile json map=map automap=reuse fileref=jsonfile noalldata;

proc copy in=jsonfile out=work;
run;
```

JSON map

```
{
  "DATASETS": [
    {
      "DSNAME": "codelists",
      "TABLEPATH": "/root/codelists",
      "VARIABLES": [
        {
          "NAME": "ordinal_root",
          "TYPE": "ORDINAL",
          "PATH": "/root"
        },
        {
          "NAME": "ordinal_codelists",
          "TYPE": "ORDINAL",
          "PATH": "/root/codelists"
        },
        {
          "NAME": "conceptId",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/conceptId",
          "CURRENT_LENGTH": 6
        },
```

```
        {
          "NAME": "definition",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/definition",
          "CURRENT_LENGTH": 57
        },
        {
          "NAME": "extensible",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/extensible",
          "CURRENT_LENGTH": 4
        },
        {
          "NAME": "name",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/name",
          "CURRENT_LENGTH": 19
        },
        {
          "NAME": "preferredTerm",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/preferredTerm",
          "CURRENT_LENGTH": 64
        },
        {
          "NAME": "submissionValue",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/submissionValue",
          "CURRENT_LENGTH": 6
        }
      ]
    },
```

```
    {
      "DSNAME": "codelists_terms",
      "TABLEPATH": "/root/codelists/terms",
      "VARIABLES": [
        {
          "NAME": "ordinal_codelists",
          "TYPE": "ORDINAL",
          "PATH": "/root/codelists"
        },
        {
          "NAME": "ordinal_terms",
          "TYPE": "ORDINAL",
          "PATH": "/root/codelists/terms"
        },
        {
          "NAME": "conceptId",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/terms/conceptId",
          "CURRENT_LENGTH": 7
        },
        {
          "NAME": "definition",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/terms/definition",
          "CURRENT_LENGTH": 153
        },
        {
          "NAME": "preferredTerm",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/terms/preferredTerm",
          "CURRENT_LENGTH": 18
        },
        {
          "NAME": "submissionValue",
          "TYPE": "CHARACTER",
          "PATH": "/root/codelists/terms/submissionValue",
          "CURRENT_LENGTH": 12
        }
      ]
    },
```

Reading JSON in SAS

- By manually editing the MAP file with a text editor we can:
  - Control the length of the variables
  - Add labels, formats or informats
  - Delete, rename or retain variables

With complex JSON files (= large number of data sets) this quickly becomes a very tedious process.

REST API Endpoint from the CDISC Library	# SAS data sets from JSON
mdr/products	16
mdr/ct/packages/sdtmct-2021-03-26	8
mdr/sdtmig/3-3	28
mdr/sdtm/1-7	39

Lua

- Lua was created in 1993 by Roberto Ierusalimschy (Rio de Janeiro)
- Lua is a modern open-source programming language with a very simple syntax
- Lua supports highly flexible data structures and modules
- Tables are the only data structure available in Lua which can be used to create different types like arrays and dictionaries
- Lua is known for excellent performance, both in speed and memory
- Lua does not replace the SAS DATA step or procedures but enhances the ability to drive SAS. Lua has direct access to the vast majority of SAS functions
- Lua makes parsing JSON very easy with one of the available modules for encoding and decoding JSON

Lua (/ˈluːə/ LOO-ə; from Portuguese: *lua* [ˈlu.(w)ɐ] meaning *moon*)

Proc LUA in SAS

- Base SAS® 9.4 introduced the LUA procedure as an alternative to the SAS Macro Language
- PROC LUA runs the Lua virtual machine inside the SAS process to offer seamless integration with SAS
- Execute Lua code within a SUBMIT / ENDSUBMIT block in PROC LUA
- Lua is a dynamically typed language – variables do not have types; only values do
- Basic types in Lua:
  nil, boolean, number, string, table, function, userdata, and thread

```
proc lua ;
  submit;

  -- Lua statements in SAS
  print('Hello world')

  endsubmit;
run;
```

Tables in Lua

- Tables are the sole data-structuring mechanism in Lua
- They can be used to represent ordinary arrays, associative arrays, lists, symbol tables, sets, records, graphs, trees, etc. -- pretty much any type of data structure in memory
- Tables can contain other tables

```lua
hours_synonyms = {'Hours', 'hr', 'h'}   -- simple array

for i, synonym in ipairs(hours_synonyms) do
  print(i, synonym)
end

terms = {}  -- associative array
terms.conceptId = "C25529"
terms.definition = "Terminology Codelist used for units within CDISC"
terms.name = "Unit"
terms.preferredTerm = "CDISC SDTM Unit of Measure Terminology"
terms.submissionValue = "UNIT"
terms.synonyms = hours_synonyms
terms.extendedValue = false
```

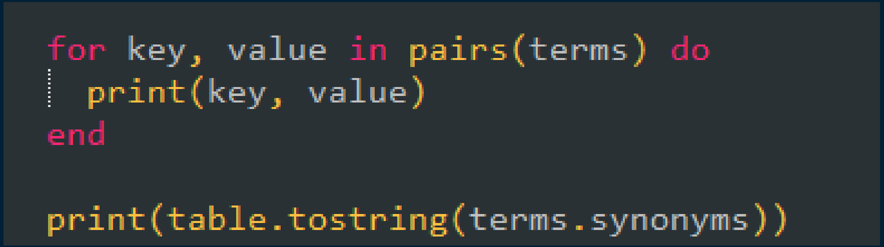
```
1   Hours
2   hr
3   h
```

Tables in Lua

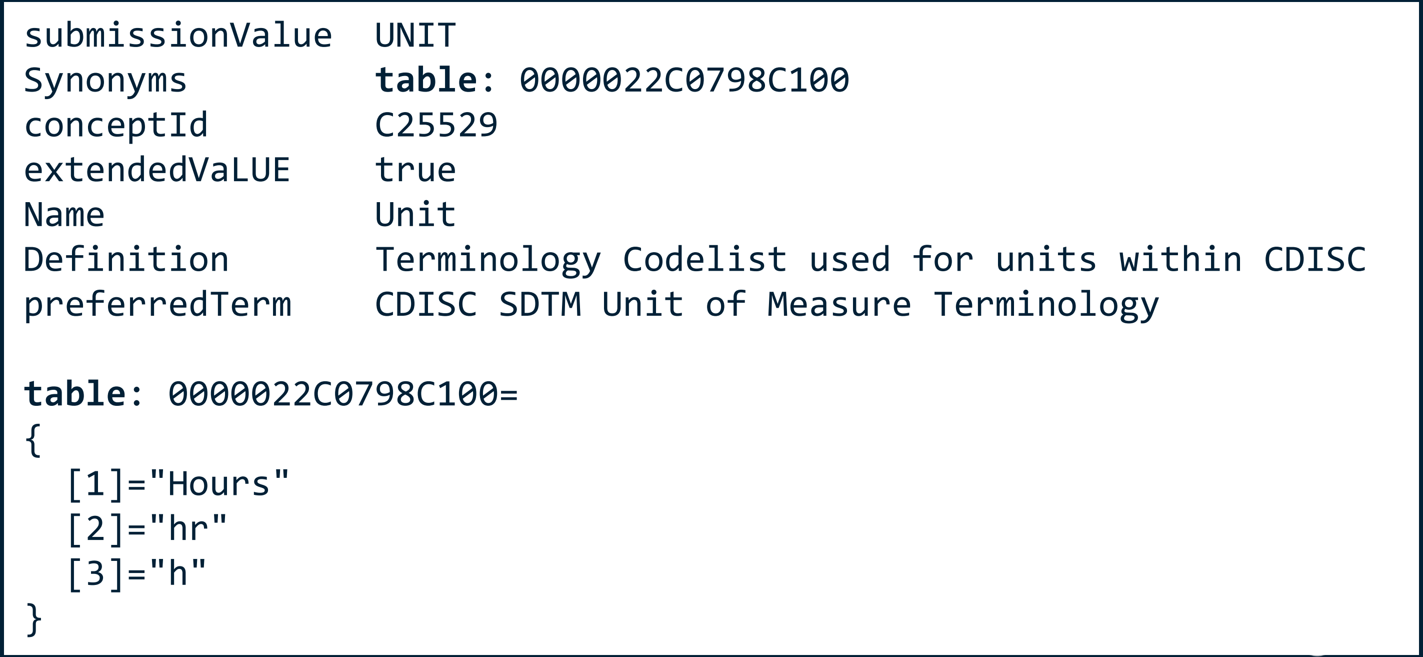
```
print(table.tostring(terms))
```

```
table: 0000022C0798BEC0=
{
  ["submissionValue"]="UNIT"
  ["synonyms"]=table: 0000022C0798C100=
  {
    [1]="Hours"
    [2]="hr"
    [3]="h"
  }
  ["conceptId"]="C25529"
  ["extendedValue"]=false
  ["name"]="Unit"
  ["definition"]="Terminology Codelist used
for units within CDISC"
  ["preferredTerm"]="CDISC SDTM Unit of
Measure Terminology"
}
```

Tables in Lua



```lua
for key, value in pairs(terms) do
  print(key, value)
end

print(table.tostring(terms.synonyms))
```



```
submissionValue  UNIT
Synonyms         table: 0000022C0798C100
conceptId        C25529
extendedVaLUE    true
Name             Unit
Definition       Terminology Codelist used for units within CDISC
preferredTerm    CDISC SDTM Unit of Measure Terminology

table: 0000022C0798C100=
{
  [1]="Hours"
  [2]="hr"
  [3]="h"
}
```

SAS and Lua – the sas table

- PROC LUA creates a special global Lua table called sas
- The sas table contains functions: `sas.scan`, `sas.symget`, `sas.symput`, ...

```
%let foo=conference;

proc lua;
  submit;
  local foo = sas.symget("foo")
  print("foo is ", foo) -- prints 'conference'
  sas.symput('foo','PHUSE')
  endsubmit;
run;

%put &foo; /* prints 'PHUSE' */
```

SAS and Lua – submitting SAS code

- PROC LUA can submit SAS code
- An optional table parameter with key-value pairs can be made available for resolution in the block of SAS code

```
local products_dataset = "prod.products"

sas.submit([[
      proc sort data=@dataset@;
        by @sort_key@;
      run;
]], { dataset=products_dataset, sort_key="product_href" })
```

```
proc sort data=prod.products;
  by product_href;
run;
```

SAS and Lua – create a SAS data set

- PROC LUA can read SAS data set and create SAS datasets
- Use the `sas.new_table` function to create a new (empty) data set template

```
function cdisclibrary.create_codelist_template(dataset_name)

  sas.new_table(dataset_name, {
      { name="codelist_name", type="C", length=256, label="Codelist Name"},
      { name="codelist_submissionValue", type="C", length=128, label="CDISC Submission Value"},
      { name="codelist_definition", type="C", length=1024, label="Codelist Definition"},
      { name="codelist_conceptId", type="C", length=8, label="Codelist Code"},
      { name="codelist_preferredTerm", type="C", length=256, label="Codelist Preferred Term"},
      { name="codelist_extensible", type="C", length=8, label="Codelist Extensible"},
      { name="term_submissionValue", type="C", length=256, label="CDISC Submission Value"},
      { name="term_conceptId", type="C", length=8, label="Term Code"},
      { name="term_synonyms", type="C", length=1024, label="Term Synonyms"},
      { name="term_definition", type="C", length=2048, label="Term Definition"},
      { name="term_preferredTerm", type="C", length=512, label="Preferred Term"}
  })

  local dsid = sas.open(dataset_name, "u")
  return dsid
end
```

SAS and Lua – create a data set

Typical scenario:

open an empty SAS data set from a template and write out observations in a loop

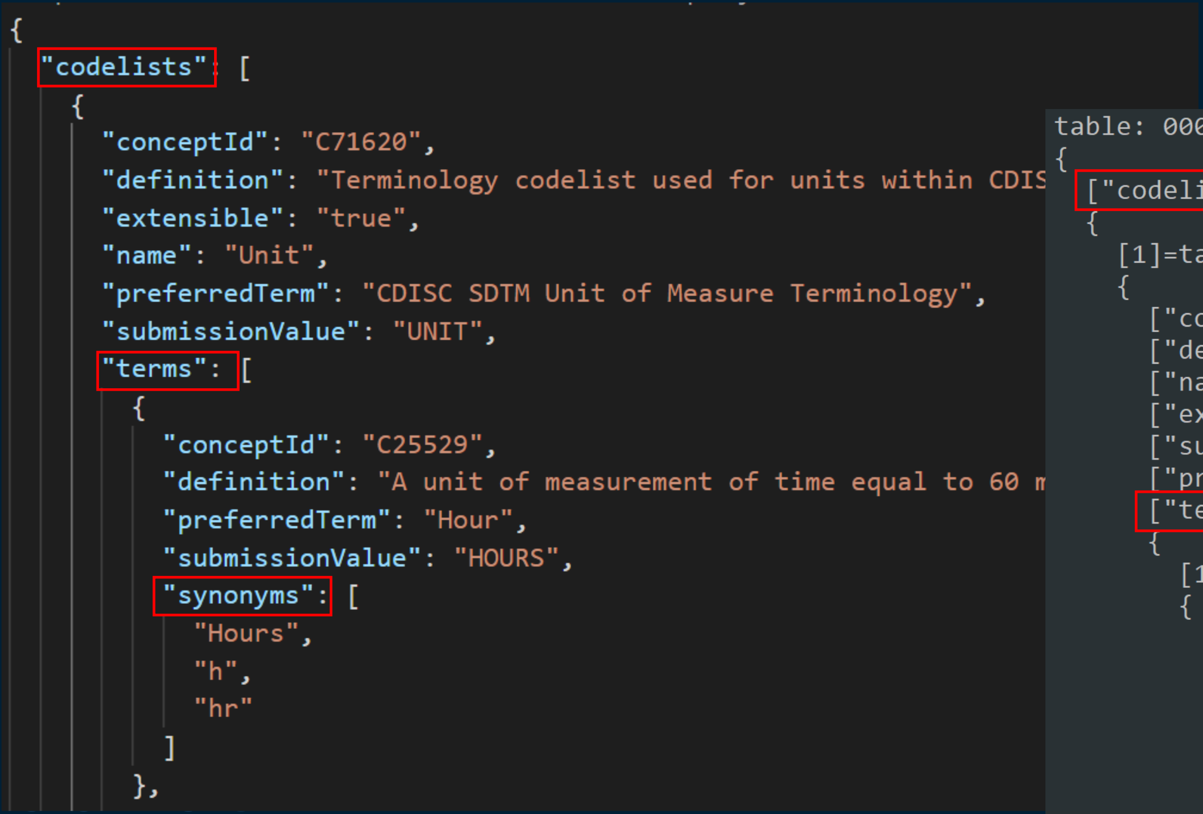
```
local dsid = cdisclibrary.create_codelist_template("out.sdtmct_20210625")

local codelists = ...    -- request JSON file and parse to a Lua table

for every codelist in codelists do
  for every term in codelist.terms do

      -- write an observation to the SAS data set
      sas.append(dsid)

      codelist_name =
      codelist_submissionValue =
      ...
      term_submissionValue =
      ...
      term_synonyms =      -- concatenate all values of term.synonyms
      ...

      sas.update(dsid)
  end
end

sas.close(dsid)
```

SAS and Lua – create a data set

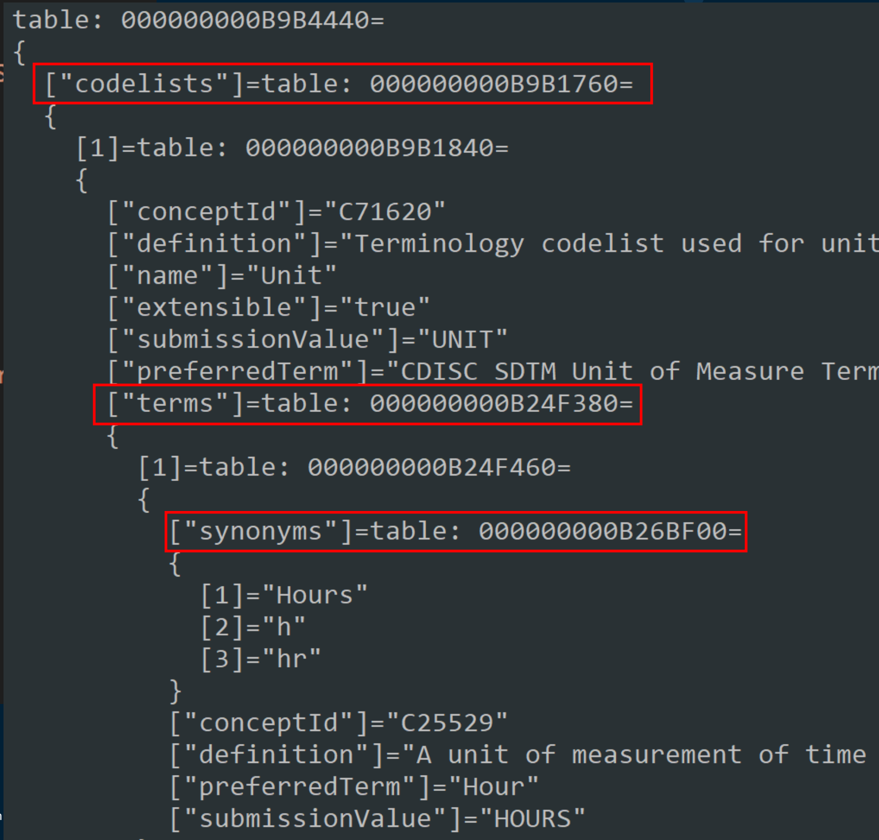
```lua
function cdisclibrary.codelists_lua2sas(dsid, lua_table)

  local codelists = lua_table.codelists
  for index, codelist in pairs(codelists) do
    local terms = codelist.terms
    if terms then
      for index2, term in pairs(terms) do
        sas.append(dsid)

        sas.put_value(dsid, "codelist_name", codelist.name)
        sas.put_value(dsid, "codelist_submissionValue", codelist.submissionValue)
        sas.put_value(dsid, "codelist_definition", codelist.definition)
        sas.put_value(dsid, "codelist_conceptId", codelist.conceptId)
        sas.put_value(dsid, "codelist_preferredTerm", codelist.preferredTerm)
        sas.put_value(dsid, "codelist_extensible", map_extensible(codelist.extensible))
        sas.put_value(dsid, "term_submissionValue", term.submissionValue)
        sas.put_value(dsid, "term_conceptId", term.conceptId)
        if term.synonyms then sas.put_value(dsid, "term_synonyms", table.concat(term.synonyms, "; ")) end
        sas.put_value(dsid, "term_preferredTerm", term.preferredTerm)
        sas.put_value(dsid, "term_definition", term.definition)

        sas.update(dsid)
      end
    end
  end
  return true
end
```

Lua and JSON

- Any JSON object can be expressed as a Lua table



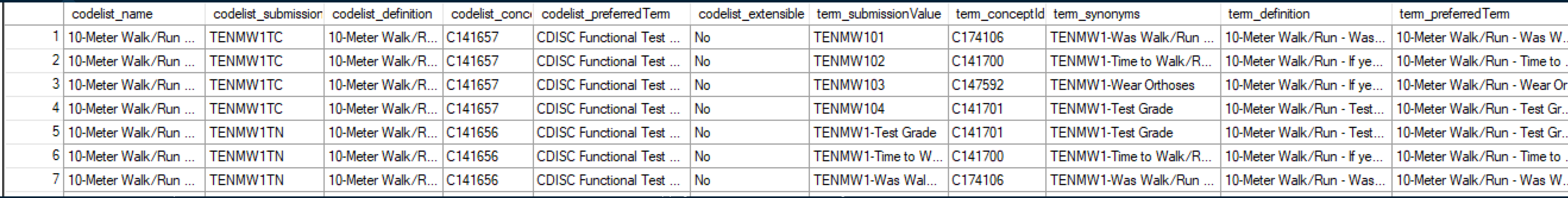
```
{
  "codelists": [
    {
      "conceptId": "C71620",
      "definition": "Terminology codelist used for units within CDIS
      "extensible": "true",
      "name": "Unit",
      "preferredTerm": "CDISC SDTM Unit of Measure Terminology",
      "submissionValue": "UNIT",
      "terms": [
        {
          "conceptId": "C25529",
          "definition": "A unit of measurement of time equal to 60 m
          "preferredTerm": "Hour",
          "submissionValue": "HOURS",
          "synonyms": [
            "Hours",
            "h",
            "hr"
          ]
        },
```



```
table: 000000000B9B4440=
{
  ["codelists"]=table: 000000000B9B1760=
  {
    [1]=table: 000000000B9B1840=
    {
      ["conceptId"]="C71620"
      ["definition"]="Terminology codelist used for unit
      ["name"]="Unit"
      ["extensible"]="true"
      ["submissionValue"]="UNIT"
      ["preferredTerm"]="CDISC SDTM Unit of Measure Term
      ["terms"]=table: 000000000B24F380=
      {
        [1]=table: 000000000B24F460=
        {
          ["synonyms"]=table: 000000000B26BF00=
          {
            [1]="Hours"
            [2]="h"
            [3]="hr"
          }
          ["conceptId"]="C25529"
          ["definition"]="A unit of measurement of time
          ["preferredTerm"]="Hour"
          ["submissionValue"]="HOURS"
```

Lua and JSON – JSON libraries

- There are a several Lua libraries available to encode a Lua table to JSON or decode JSON to a Lua table (http://lua-users.org/wiki/JsonModules)
  This site reviews speed, bugs and edge cases
- I tested some of the (pure) Lua implementations
- Test file: CDISC Library SDTM Controlled Terminology 2021-06-25 sdtmct_20210625.json, 892 code lists with a total of 32570 terms and a total of 28276 term synonyms.
- The goal was to create the same data set with the SAS JSON libname engine and several JSON Lua libraries.



	codelist_name	codelist_submission	codelist_definition	codelist_conc	codelist_preferredTerm	codelist_extensible	term_submissionValue	term_conceptId	term_synonyms	term_definition	term_preferredTerm
1	10-Meter Walk/Run ...	TENMW1TC	10-Meter Walk/R...	C141657	CDISC Functional Test ...	No	TENMW101	C174106	TENMW1-Was Walk/Run ...	10-Meter Walk/Run - Was...	10-Meter Walk/Run - Was W
2	10-Meter Walk/Run ...	TENMW1TC	10-Meter Walk/R...	C141657	CDISC Functional Test ...	No	TENMW102	C141700	TENMW1-Time to Walk/R...	10-Meter Walk/Run - If ye...	10-Meter Walk/Run - Time to
3	10-Meter Walk/Run ...	TENMW1TC	10-Meter Walk/R...	C141657	CDISC Functional Test ...	No	TENMW103	C147592	TENMW1-Wear Orthoses	10-Meter Walk/Run - If ye...	10-Meter Walk/Run - Wear Or
4	10-Meter Walk/Run ...	TENMW1TC	10-Meter Walk/R...	C141657	CDISC Functional Test ...	No	TENMW104	C141701	TENMW1-Test Grade	10-Meter Walk/Run - Test...	10-Meter Walk/Run - Test Gr
5	10-Meter Walk/Run ...	TENMW1TN	10-Meter Walk/R...	C141656	CDISC Functional Test ...	No	TENMW1-Test Grade	C141701	TENMW1-Test Grade	10-Meter Walk/Run - Test...	10-Meter Walk/Run - Test Gr
6	10-Meter Walk/Run ...	TENMW1TN	10-Meter Walk/R...	C141656	CDISC Functional Test ...	No	TENMW1-Time to W...	C141700	TENMW1-Time to Walk/R...	10-Meter Walk/Run - If ye...	10-Meter Walk/Run - Time to
7	10-Meter Walk/Run ...	TENMW1TN	10-Meter Walk/R...	C141656	CDISC Functional Test ...	No	TENMW1-Was Wal...	C174106	TENMW1-Was Walk/Run ...	10-Meter Walk/Run - Was...	10-Meter Walk/Run - Was W

Lua and JSON – JSON libraries

- [jf-JSON] Jeffrey Friedl's Lua module for encoding and decoding JSON in Lua
  http://regex.info/blog/lua/json – Version 20170927.26 (2017)
  Creative Commons Attribution 3.0 Unported License
- [dkjson] David Kolf's JSON Module for Lua - http://dkolf.de/src/dkjson-lua.fsl/home
  Version 2.5 (2014-04-28) - MIT/X11 license
- [dkjson-wiki] Fandom Developers Wiki: JSON high-performance bidirectional conversion framework. This module is a fork of the dkjson library by David Kolf
  https://dev.fandom.com/wiki/Global_Lua_Modules/Json – Version 2.5.0+wikia:dev (2020)
  MIT license
- [rxi-json] Json.lua - A lightweight JSON library for Lua - https://github.com/rxi/json.lua
  Version 0.1.2 (2019) – MIT license
- [luna-json] Lunajson features a SAX-style JSON parser and simple JSON decoder/encoder.
  https://github.com/grafi-tt/lunajson - Version 1.2.3 (2020) - MIT/X11 license

JSON to Lua to SAS – complete example

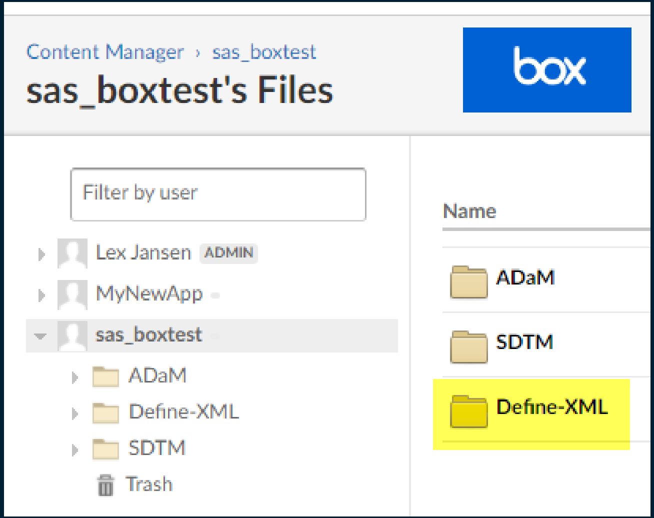
```
filename luapath ("./lua" "./lua/jsonlibraries");

proc lua restart;
  submit;

    local fileutils = require 'fileutils'
    local rest = require 'rest'
    local cdisclibrary = require 'cdisclibrary'
    local jf_json = require 'jsonlibraries.jf_json'

    sas.gfilename('jsonfile', 'sdtmct_20210625.json')

    rest.base_url = 'https://library.cdisc.org/api'
    local token = '[redacted]'
    rest.headers='"Accept"="application/json" "api-key"='..'"'..token..'"'

    local pass,code = rest.request('get', '/mdr/ct/packages/sdtmct-2021-06-25', 'jsonfile')

    local json_string = fileutils.read('jsonfile')
    local json_table = jf_json:decode(json_string)

    local dsid = cdisclibrary.create_codelist_template('work.sdtmct_20210625')
    cdisclibrary.codelists_lua2sas(dsid, json_table)
    sas.close(dsid)

  endsubmit;
run;
```

Lua and JSON – JSON libraries

- Speed

Method	Real time used
SAS JSON libname engine - PROC COPY + SQL joins + data step	0.32 + 0.57 + 0.13 = 1.42 sec.
jf-JSON Lua library	3.42 sec.
dkjson Lua library	1.51 sec.
dkjson-wiki Lua library	1.53 sec.
rxi-json Lua library	1.78 sec.
luna-json Lua library	1.07 sec.

JSON to Lua to SAS – Box.com example

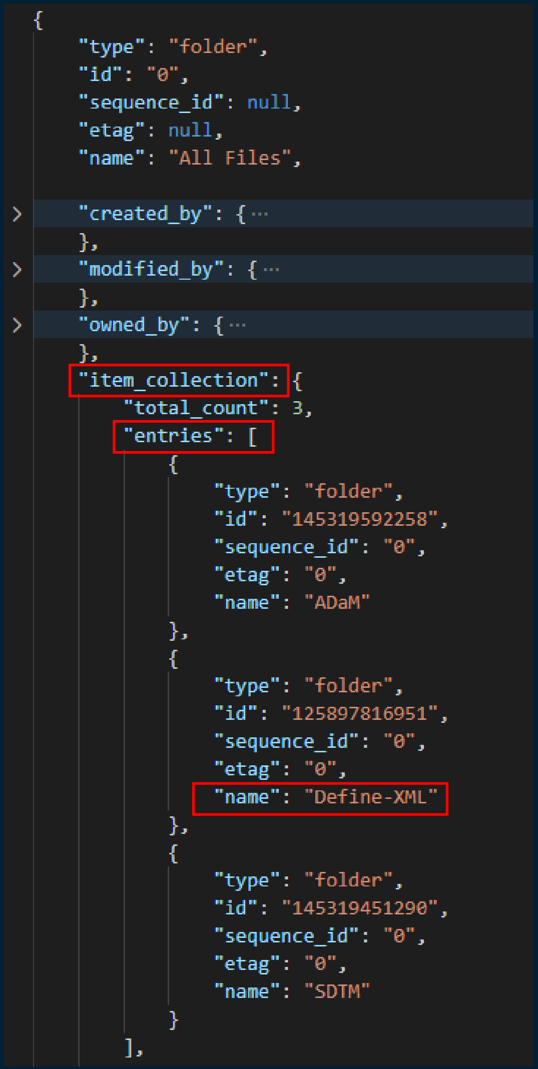


```
filename response "response_folder_check.json";
filename resphdrs temp;

proc http
  url="https://api.box.com/2.0/folders/0"
  method = "GET"
  out = response
  headerout = resphdrs
  ct = "application/json";
  headers
    "Authorization" = "Bearer [redacted]";
run;
```

- You want to check if the folder "Define-XML" exists in your **box.com** account

This will return a JSON file

JSON to Lua to SAS – Box.com example



```
{
    "type": "folder",
    "id": "0",
    "sequence_id": null,
    "etag": null,
    "name": "All Files",

    "created_by": {…
    },
    "modified_by": {…
    },
    "owned_by": {…
    },
    "item_collection": {
        "total_count": 3,
        "entries": [
            {
                "type": "folder",
                "id": "145319592258",
                "sequence_id": "0",
                "etag": "0",
                "name": "ADaM"
            },
            {
                "type": "folder",
                "id": "125897816951",
                "sequence_id": "0",
                "etag": "0",
                "name": "Define-XML"
            },
            {
                "type": "folder",
                "id": "145319451290",
                "sequence_id": "0",
                "etag": "0",
                "name": "SDTM"
            }
        ],
```

```
%let boxfolder=Define-XML;
%let BoxFolderExists=0;
%let FolderID=;

proc lua;
  submit;
    local response_json = json:decode(fileutils.read('response'))
    sas.symput('folderID', response_json['id']);
    local t = response_json['item_collection']['entries']
    for key, value in pairs(t) do
      if value.name == sas.symget('boxfolder') then
        sas.symput('BoxFolderExists', 1)
        sas.symput('FolderID', value.id)
      end
    end
  endsubmit;
run;

%put &=BoxFolderExists (Folder=&boxfolder, &=FolderID);

%if (not &BoxFolderExists) %then
  %do;

    ...... code to create folder

  %end;
```

```
156
157         %put &=BoxFolderExists (Folder=&boxfolder, &=FolderID);
BOXFOLDEREXISTS=1 (Folder=Define-XML, FOLDERID=125897816951)
158
```

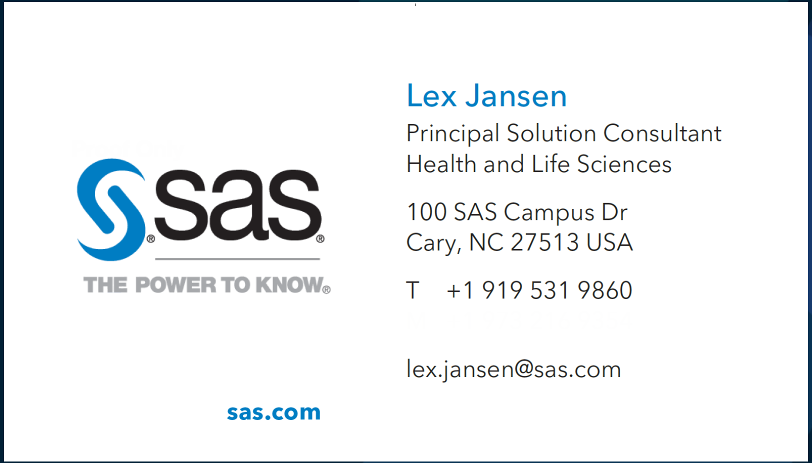
Conclusion

- PROC LUA greatly enhances SAS/Base capabilities for converting complex JSON files into SAS data sets

References & Suggested Reading

- The JavaScript Object Notation (JSON) Data Interchange Format, RFC 8259
  T. Bray, Ed., Internet Engineering Task Force (IETF), December 2017
- The JSON Data Interchange Format, Standard ECMA-404
  Ecma International, 2nd edition, December 2017
- Driving SAS® with Lua
  Paul Tomas, SAS Global Forum 2015
- REST Easier with SAS®: Using the LUA Procedure to Simplify REST API Interactions
  Steven Major, SAS Global Forum 2019 (code on GitHub)
- The Programming Language Lua (http://www.lua.org)
  Lua.org (2013)

Thank You !
Questions ?



Lex Jansen
Principal Solution Consultant
Health and Life Sciences

100 SAS Campus Dr
Cary, NC 27513 USA

T +1 919 531 9860

lex.jansen@sas.com

sas.com

All code can be found at GitHub:
https://github.com/lexjansen/sas-papers/tree/master/phuse_eu-2021