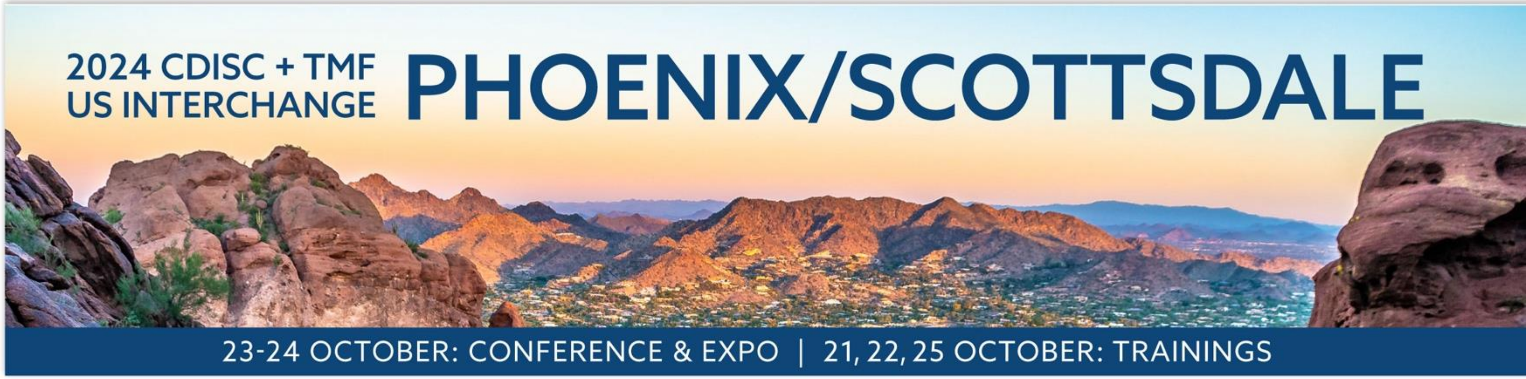

2024 CDISC + TMF US INTERCHANGE
PHOENIX/SCOTTSDALE
23-24 OCTOBER: CONFERENCE & EXPO | 21, 22, 25 OCTOBER: TRAININGS

Running the CDISC Open Rules Engine (CORE) in BASE SAS©

Presented by Lex Jansen, Senior Director, Data Science Development, CDISC

Meet the Speaker

Lex Jansen

Title: Senior Director, Data Science Development

Organization: CDISC

Lex Jansen is an independent consultant, currently working as Senior Director, Data Science Development at CDISC. Lex co-leads the CDISC Data Exchange Standards team, and contributes to the CDISC Library and the CDISC Biomedical Concepts project.
Before joining CDISC, he was a Principal Solution Consultant and Principal Software Developer at SAS Institute.

Disclaimer and Disclosures

- *The views and opinions expressed in this presentation are those of the author(s) and do not necessarily reflect the official policy or position of CDISC.*
- *CDISC is a vendor-neutral and technology-inclusive organization focused on promoting the use of standards to improve the quality and efficiency of research*
- *CDISC does not endorse any specific vendor or technology in the use of its standards.*
- *The author has no conflicts to disclose*

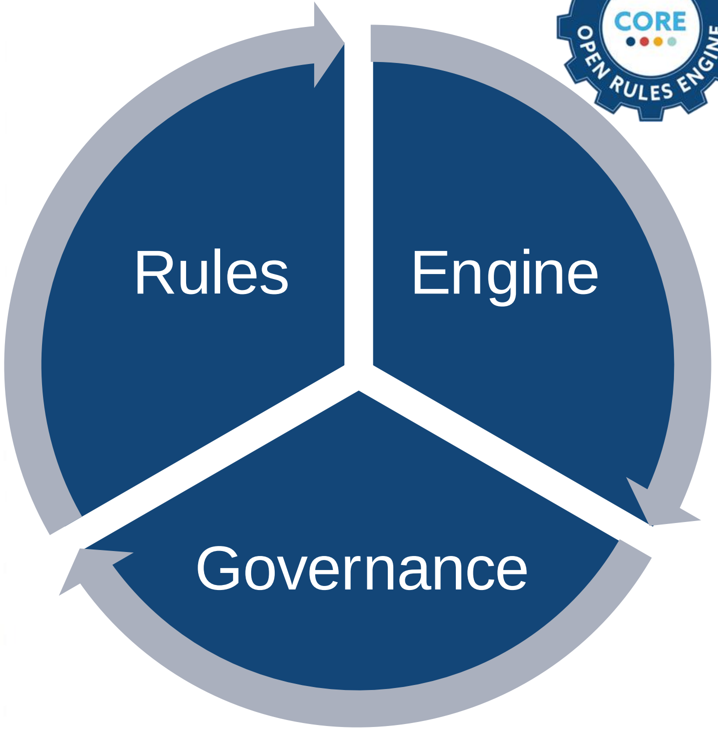
Agenda

1. Introduction - CORE Concept
2. CORE Conformance Rules
3. The CORE Engine
4. Implementing the CORE engine in SAS
5. Running the CORE engine in SAS

Introduction

The CORE Concept

What are CDISC Open Rules?

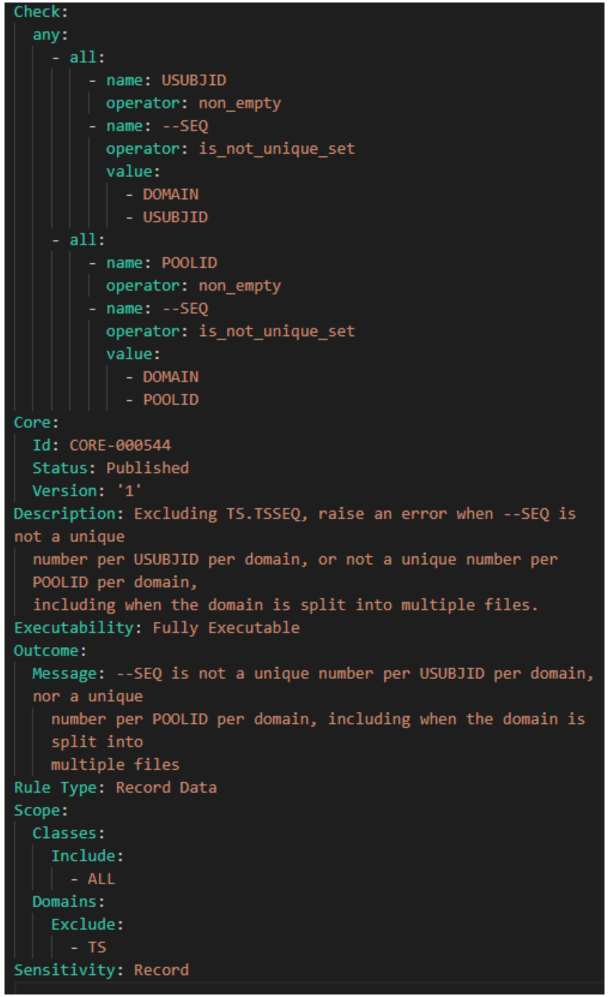


- **Rules:** Complete set of aligned, open and unambiguous machine-readable conformance rules for each standard including CDISC, Regulatory, and Industry needs
- **Governance:** Well-defined governance model for the evaluation, development, and publication of rules from all stakeholders
- **Engine:** Open-source rules engine available for testing and community use

CORE Conformance Rules

CORE Conformance Rules

- A human-readable Rule specification is interpreted by the Rule developer and authored in the CORE Rule Editor using a structured language (YAML).
- Rule Editor:
    - Web-based application
    - Structured document (YAML), 1 CORE rule per file containing rule's metadata and check logic
    - Real-time syntax checking



```
Check:
  any:
    - all:
        - name: USUBJID
          operator: non_empty
        - name: --SEQ
          operator: is_not_unique_set
          value:
            - DOMAIN
            - USUBJID
    - all:
        - name: POOLID
          operator: non_empty
        - name: --SEQ
          operator: is_not_unique_set
          value:
            - DOMAIN
            - POOLID
Core:
  Id: CORE-000544
  Status: Published
  Version: '1'
Description: Excluding TS.TSSEQ, raise an error when --SEQ is
not a unique
  number per USUBJID per domain, or not a unique number per
  POOLID per domain,
  including when the domain is split into multiple files.
Executability: Fully Executable
Outcome:
  Message: --SEQ is not a unique number per USUBJID per domain,
  nor a unique
    number per POOLID per domain, including when the domain is
    split into
    multiple files
Rule Type: Record Data
Scope:
  Classes:
    Include:
      - ALL
  Domains:
    Exclude:
      - TS
Sensitivity: Record
```

The CORE Engine

The CORE Engine

- Open-source software application whose purpose is to execute the Rules against clinical data and return results.
- The CORE Engine is made available to the CDISC Community in GitHub (https://github.com/cdisc-org/cdisc-rules-engine)
- The Engine is written in the Python programming language and comes with a permissive MIT open-source license
- Can be deployed in a variety of processing environments
- The Engine accesses the Rules from the CDISC Library via a Library API when it executes
- Users may also add custom Rules for processing

The CORE Engine

There are several ways to run the CORE Engine

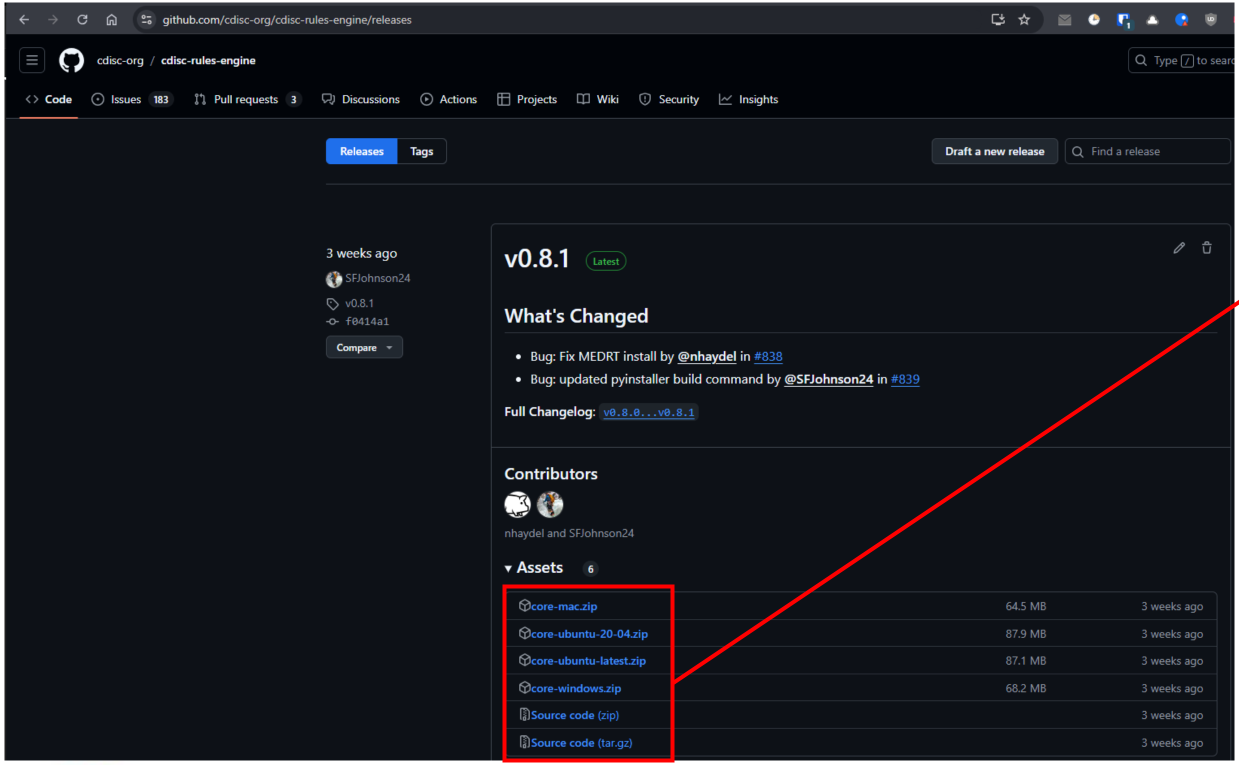
- As a CLI (Command Line Interface)
  - compiled packages available for Windows, Mac, Linux-Ubuntu
  - Download - Unzip - Run
  - `.\core.exe validate -s <standard> -v <standard_version> -d <datasetpath>`
    `.\core.exe validate -s sdtmig -v 3-4 -d .\data\`

- Run in Python
  - Clone the repository and run `python core.py` from the root of the CORE project with appropriate parameters.

- Import the rules engine library in Python (available as a package on PyPi) and run rules against data directly (without needing your data to be in .xpt format) in your own environment or tooling

The CORE Engine - Running as a CLI

https://github.com/cdisc-org/cdisc-rules-engine/releases



3 weeks ago
SFJohnson24
v0.8.1
f0414a1

v0.8.1 Latest

What's Changed

- Bug: Fix MEDRT install by @nhaydel in #838
- Bug: updated pyinstaller build command by @SFJohnson24 in #839

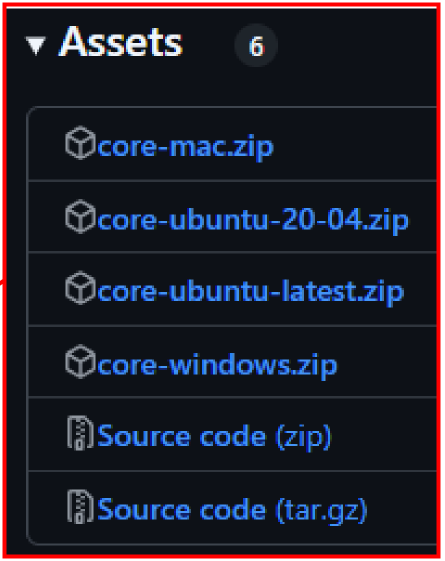
Full Changelog: v0.8.0...v0.8.1

Contributors

nhaydel and SFJohnson24

Assets 6

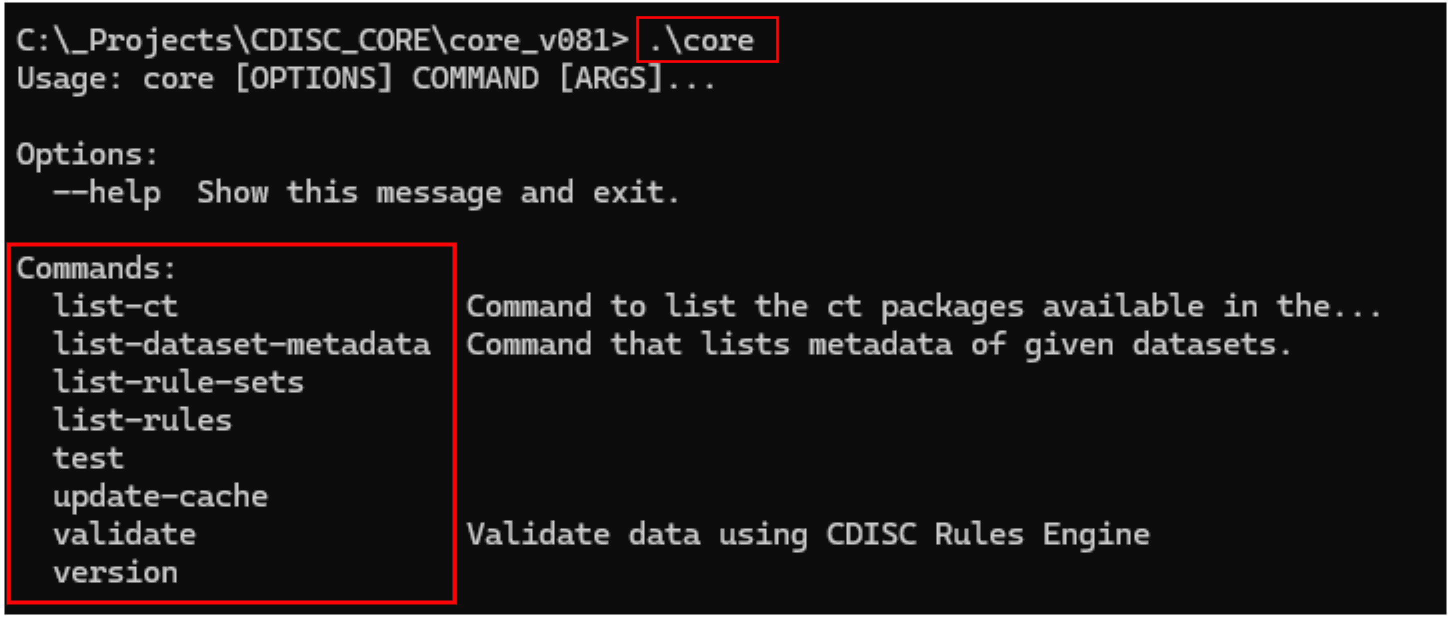
core-mac.zip	64.5 MB	3 weeks ago
core-ubuntu-20-04.zip	87.9 MB	3 weeks ago
core-ubuntu-latest.zip	87.1 MB	3 weeks ago
core-windows.zip	68.2 MB	3 weeks ago
Source code (zip)		3 weeks ago
Source code (tar.gz)		3 weeks ago



Assets 6

- core-mac.zip
- core-ubuntu-20-04.zip
- core-ubuntu-latest.zip
- core-windows.zip
- Source code (zip)
- Source code (tar.gz)

The CORE Engine - Running as a CLI

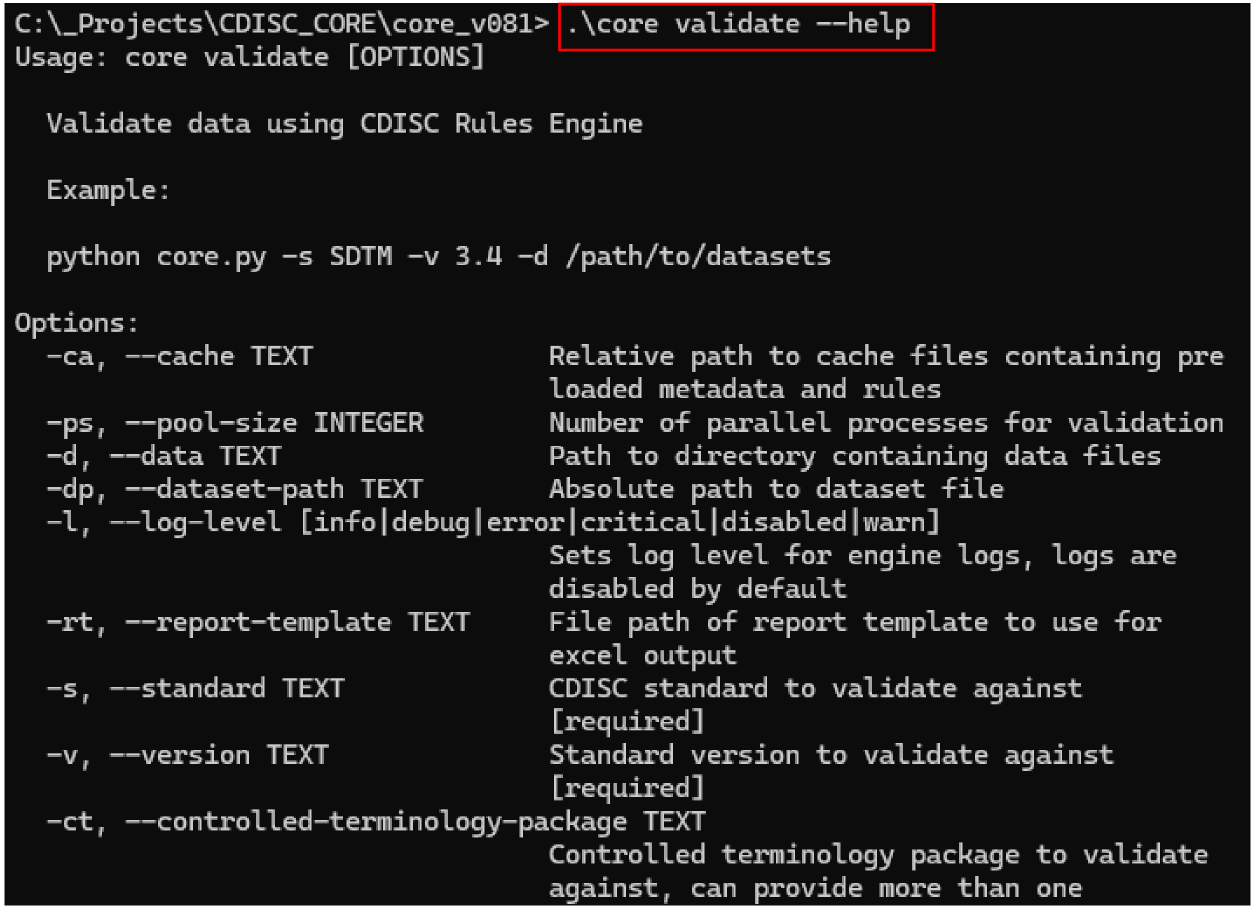


```
C:\_Projects\CDISC_CORE\core_v081> .\core
Usage: core [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  list-ct                 Command to list the ct packages available in the...
  list-dataset-metadata   Command that lists metadata of given datasets.
  list-rule-sets
  list-rules
  test
  update-cache
  validate                Validate data using CDISC Rules Engine
  version
```

The CORE Engine - Running as a CLI - Commands

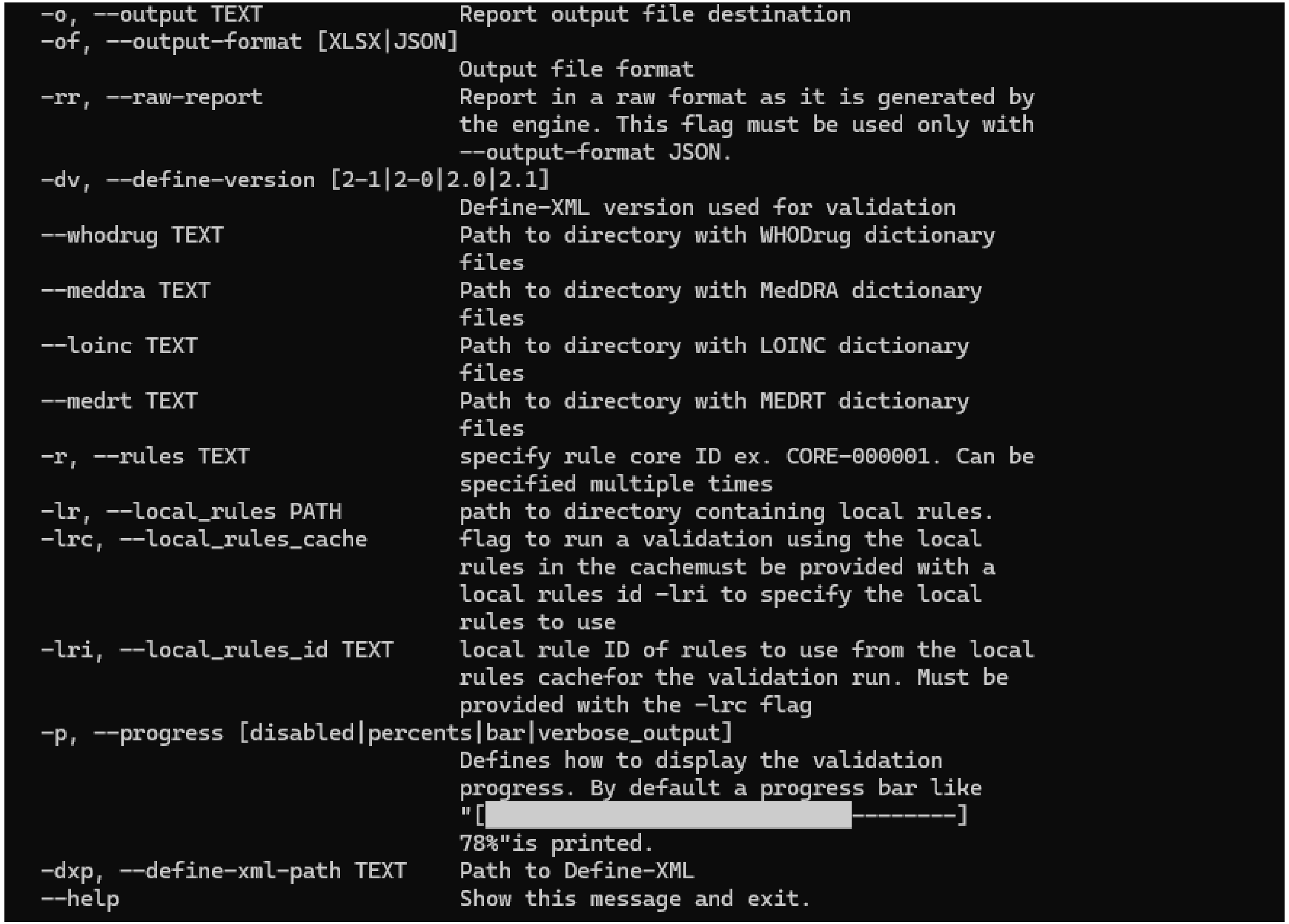
- **list-ct** - List the Controlled Terminology packages available in the cache
- **list-dataset-metadata** - Lists metadata of given datasets
- **list-rule-sets** - Lists rules sets available in the cache
- **test** - Test Rules using the CDISC Rules Engine
- **update-cache** - Update the local cache folder
- **validate** - Validate data using CDISC Rules Engine
- **version** - Show the version of the CDISC Rules Engine

The CORE Engine - Running as a CLI - validate



```
C:\_Projects\CDISC_CORE\core_v081> .\core validate --help
Usage: core validate [OPTIONS]

  Validate data using CDISC Rules Engine

  Example:

  python core.py -s SDTM -v 3.4 -d /path/to/datasets

Options:
  -ca, --cache TEXT                Relative path to cache files containing pre
                                   loaded metadata and rules
  -ps, --pool-size INTEGER         Number of parallel processes for validation
  -d, --data TEXT                  Path to directory containing data files
  -dp, --dataset-path TEXT         Absolute path to dataset file
  -l, --log-level [info|debug|error|critical|disabled|warn]
                                   Sets log level for engine logs, logs are
                                   disabled by default
  -rt, --report-template TEXT      File path of report template to use for
                                   excel output
  -s, --standard TEXT              CDISC standard to validate against
                                   [required]
  -v, --version TEXT               Standard version to validate against
                                   [required]
  -ct, --controlled-terminology-package TEXT
                                   Controlled terminology package to validate
                                   against, can provide more than one
```

The CORE Engine - Running as a CLI - validate

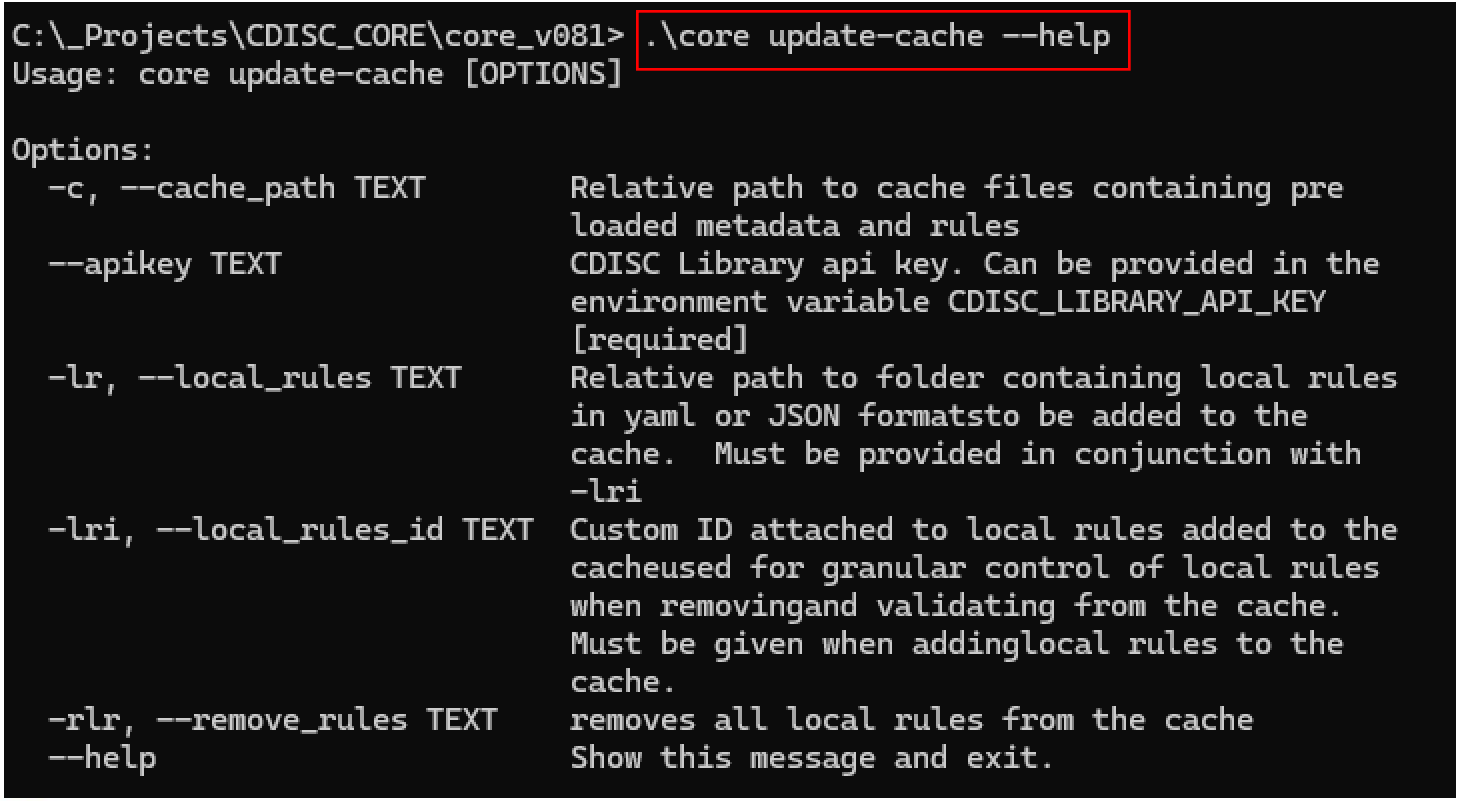


```
  -o, --output TEXT               Report output file destination
  -of, --output-format [XLSX|JSON]
                                  Output file format
  -rr, --raw-report               Report in a raw format as it is generated by
                                  the engine. This flag must be used only with
                                  --output-format JSON.
  -dv, --define-version [2-1|2-0|2.0|2.1]
                                  Define-XML version used for validation
  --whodrug TEXT                  Path to directory with WHODrug dictionary
                                  files
  --meddra TEXT                   Path to directory with MedDRA dictionary
                                  files
  --loinc TEXT                    Path to directory with LOINC dictionary
                                  files
  --medrt TEXT                    Path to directory with MEDRT dictionary
                                  files
  -r, --rules TEXT                specify rule core ID ex. CORE-000001. Can be
                                  specified multiple times
  -lr, --local_rules PATH         path to directory containing local rules.
  -lrc, --local_rules_cache       flag to run a validation using the local
                                  rules in the cachemust be provided with a
                                  local rules id -lri to specify the local
                                  rules to use
  -lri, --local_rules_id TEXT     local rule ID of rules to use from the local
                                  rules cachefor the validation run. Must be
                                  provided with the -lrc flag
  -p, --progress [disabled|percents|bar|verbose_output]
                                  Defines how to display the validation
                                  progress. By default a progress bar like
                                  "[████████████████████████████--------]
                                  78%"is printed.
  -dxp, --define-xml-path TEXT    Path to Define-XML
  --help                          Show this message and exit.
```

The CORE Engine - Running as a CLI - update-cache

- The CORE Engine stores rules and standards metadata from the CDISC Library in a local cache folder.
- Rules get added to the CDISC Library on a regular basis
- At any moment in time, the locally stored cache can be updated with the `update-cache` command

The CORE Engine - Running as a CLI - update-cache



```
C:\_Projects\CDISC_CORE\core_v081> .\core update-cache --help
Usage: core update-cache [OPTIONS]

Options:
  -c, --cache_path TEXT       Relative path to cache files containing pre
                              loaded metadata and rules
  --apikey TEXT               CDISC Library api key. Can be provided in the
                              environment variable CDISC_LIBRARY_API_KEY
                              [required]
  -lr, --local_rules TEXT     Relative path to folder containing local rules
                              in yaml or JSON formatsto be added to the
                              cache.  Must be provided in conjunction with
                              -lri
  -lri, --local_rules_id TEXT  Custom ID attached to local rules added to the
                              cacheused for granular control of local rules
                              when removingand validating from the cache.
                              Must be given when addinglocal rules to the
                              cache.
  -rlr, --remove_rules TEXT   removes all local rules from the cache
  --help                      Show this message and exit.
```

Implementing the CORE Engine in SAS

Running the CORE Engine in SAS

SAS has various techniques to execute commands

- X statement
- SYSTASK statement
- %SYSEXEC statement
- CALL SYSTEM statement
- SYSTEM function
- FILENAMEC statement with the PIPE option

Relevant SAS options:

- **XSYNC** - Controls whether an X command or statement executes synchronously or asynchronously
- **XWAIT** - Specifies whether you must type EXIT at the command prompt before the shell closes

Running the CORE Engine in SAS

```
%let project_folder = /_github/lexjansen/cdisc-core-sas;
%let core_exe = \_Projects\CDISC_CORE\core_v081\core.exe;
%let core_log = %sysfunc(pathname(work))/core;
%let core_options =
    -ca &project_folder/resources/cache
    -dp &project_folder/testdata/sdtm
    -rt &project_folder/resources/templates/report-template.xlsx
    -s sdtmig -v 3-3
    -ct sdtmct-2023-12-15
    -dv 2.1
    -o &project_folder/develop/sdtmig-3-3-report
    --whodrug &project_folder/testdata/dictionaries/whodrug
    --meddra &project_folder/testdata/dictionaries/meddra
    -r CORE-000266 -r CORE-000356;

systask command "&core_exe validate &core_options" wait
        taskname=core_task_validate status=core_result_validate;
%put &=core_result_validate;
```

Running the CORE Engine in SAS

It works:

```
46   systask command "&core_exe validate &core_options"
47              wait taskname=core_task_validate status=core_result_validate;
NOTE: Task "core_task_validate" produced no LOG/Output.
48   %put &=core_result_validate;
CORE_RESULT_VALIDATE=0
```

... or it does not work:

```
46   Systask command "&core_exe validate &core_options"
ERROR: Insufficient authorization to access SYSTASK COMMAND.
47  wait taskname=core_task_validate status=core_result_validate;
```

Running the CORE Engine in SAS

- The assumption is that the SYSTASK command is valid in the current SAS session.
- This may not be the case especially in shared SAS environments.
- In certain SAS environments SAS administrators do not allow command line execution using Base/SAS

- Some users have found ways around this in their SAS environment:
  - Write Java code that can execute the CORE commands.
    This Java code can be compiled into an executable jar and wrapped into a SAS macro to support CORE execution via SAS
  - In some SAS environments R can be executed, and has not been locked down.
    Write an R script that can execute the CORE commands
- Since CORE is written in Python, and SAS supports execution of Python code, why not call the Python CORE functions directly in SAS?

Running the CORE Engine in SAS

Solution:

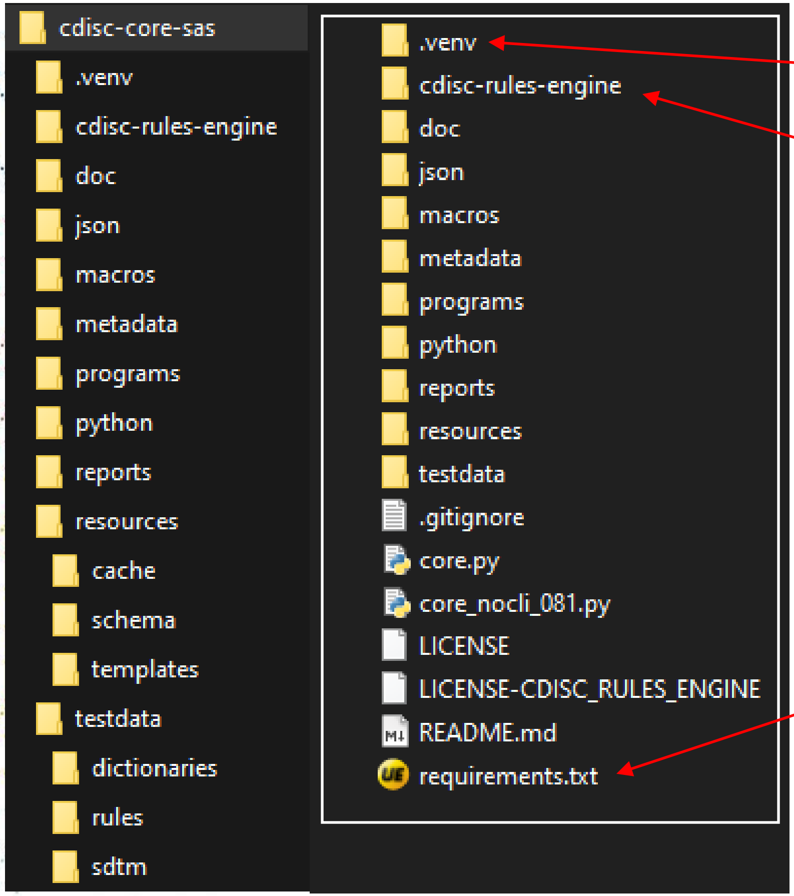
- Implement the CDISC CORE CLI commands as Python functions extracted from the CORE Python entry point (core.py)
- Pass parameters and code to the Python interpreter and return the results to SAS
- These Python functions can be called and executed by user-defined SAS functions that are defined in the SAS Function Compiler (PROC FCMP)
- These user-defined SAS functions can be called from the DATA step or any context where SAS functions are available.
- Wrap the user-defined SAS functions in SAS macro to work around FCMP limitations:
  - define named parameters
  - optional parameters
  - default parameter values

Running the CORE Engine in SAS - via Python

- Details of a Proof of Concept on GitHub
  https://github.com/lexjansen/cdisc-core-sas

- SAS 9.4M6 (May 2019 update) or later
- Python installed -
  The CDISC CORE engine requires Python 3.9 or Python 3.10
- Set the **MAS_M2PATH** and **MAS_PYPATH** environment variables
  - MAS_M2PATH - absolute path to mas2py.py file in your SAS installation
  - MAS_PYPATH - absolute path to the Python executable
- The **CORE_PATH** environment variable needs have the absolute path to a clone of the cdisc-rules-engine GitHub repository
- The cdisc-core-sas repository (https://github.com/lexjansen/cdisc-core-sas) comes bundled with the source code of the v0.8.1 release (September 24, 2024) of the CDISC CORE engine

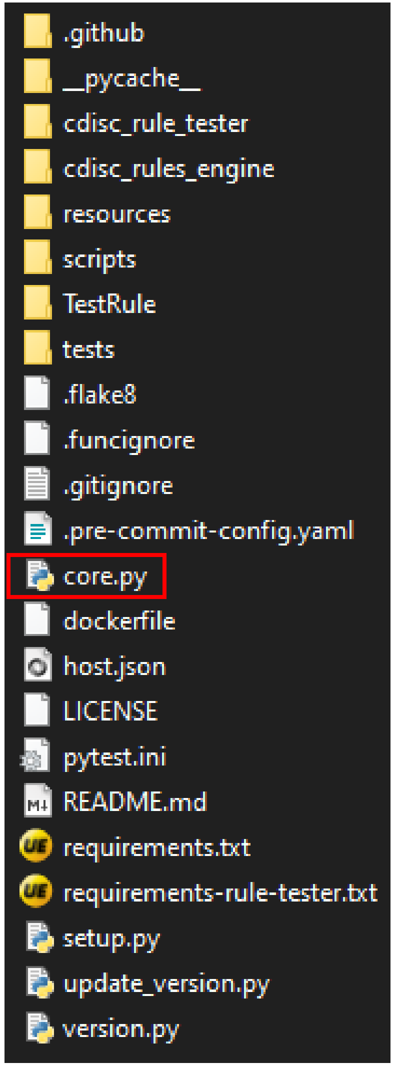
Running the CORE Engine in SAS - via Python



virtual Python environment

Source code from the v0.8.1 cdisc-rules-engine release

Requirements file with Python packages to be installed


.github
__pycache__
cdisc_rule_tester
cdisc_rules_engine
resources
scripts
TestRule
tests
.flake8
.funcignore
.gitignore
.pre-commit-config.yaml
core.py
dockerfile
host.json
LICENSE
pytest.ini
README.md
requirements.txt
requirements-rule-tester.txt
setup.py
update_version.py
version.py

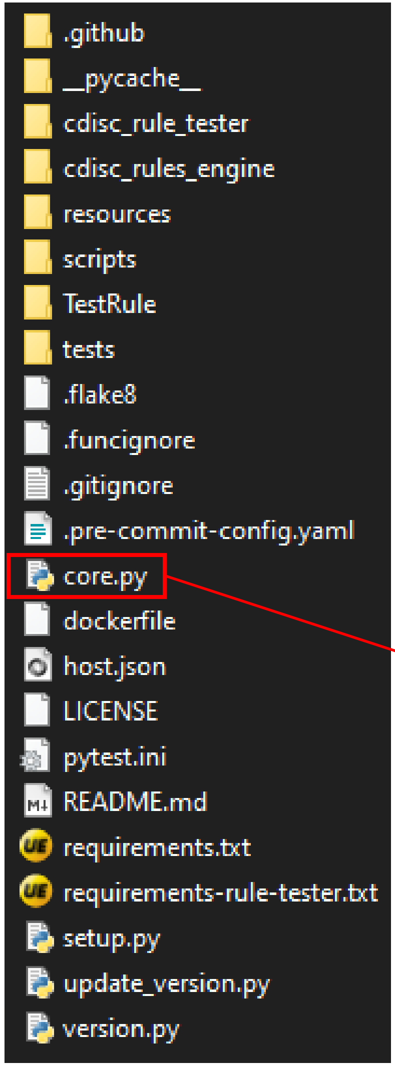
Running the CORE Engine in SAS - via Python

core.py is the interface to the CORE Engine commands

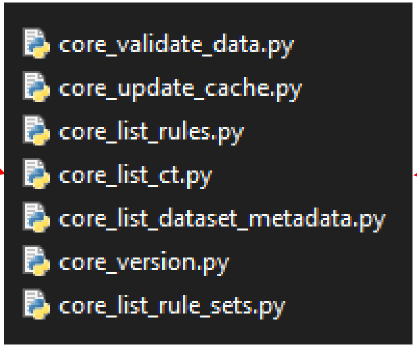
- Contains the definitions of the CORE commands
- Contains a Python function to be called for each CORE command
- Defines the parameters for the commands, including defaults and required/optional

Running the CORE Engine in SAS - via Python

From **core.py** create Python functions that can be called and executed by user-defined SAS functions, which will be called by SAS macros

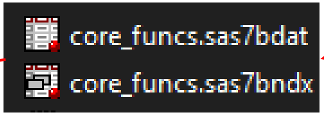


.github
__pycache__
cdisc_rule_tester
cdisc_rules_engine
resources
scripts
TestRule
tests
.flake8
.funcignore
.gitignore
.pre-commit-config.yaml
core.py
dockerfile
host.json
LICENSE
pytest.ini
README.md
requirements.txt
requirements-rule-tester.txt
setup.py
update_version.py
version.py



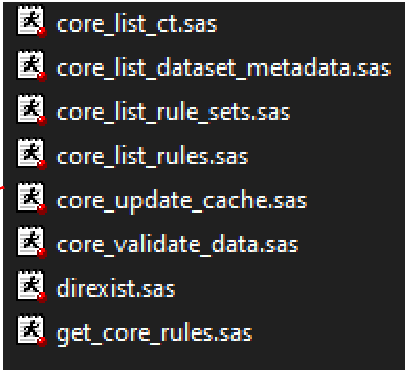
core_validate_data.py
core_update_cache.py
core_list_rules.py
core_list_ct.py
core_list_dataset_metadata.py
core_version.py
core_list_rule_sets.py

Python functions



core_funcs.sas7bdat
core_funcs.sas7bndx

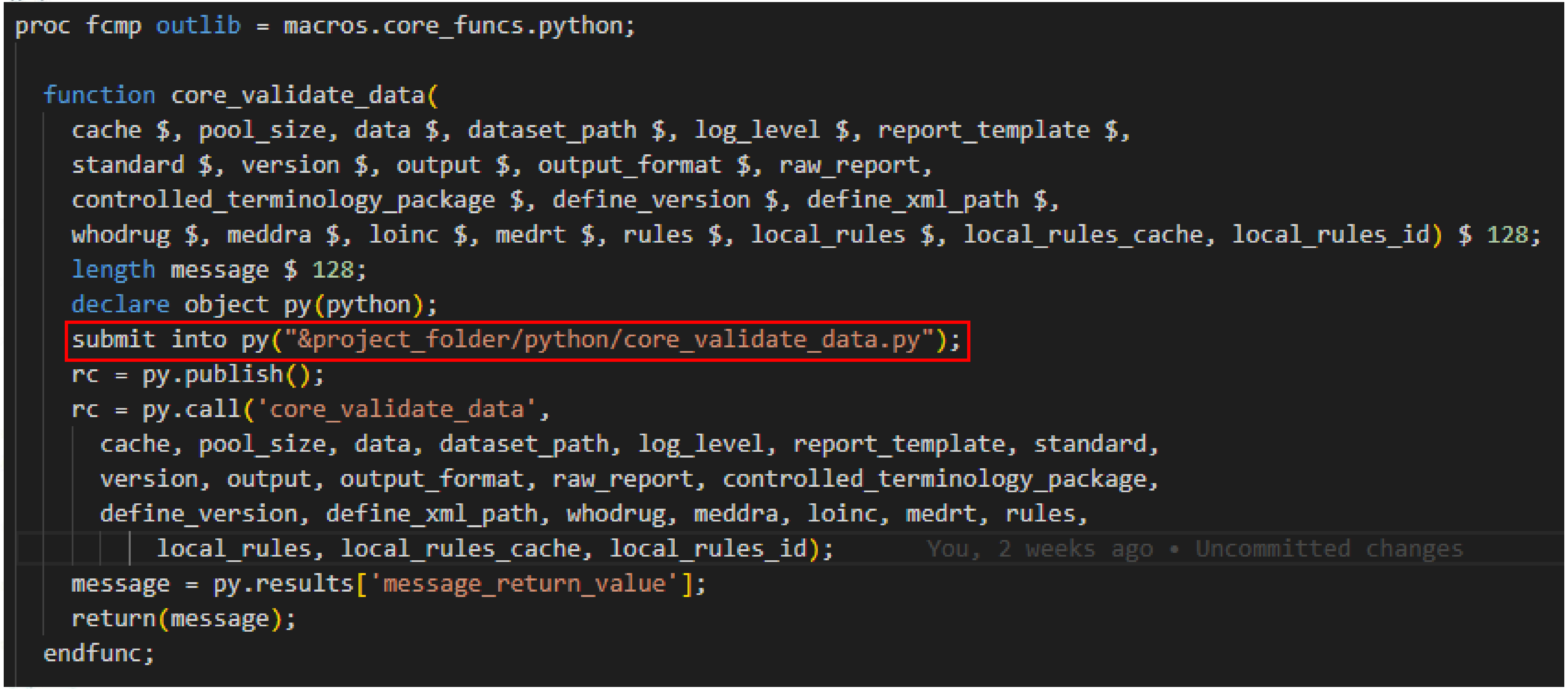
PROC FCMP user-defined SAS functions



core_list_ct.sas
core_list_dataset_metadata.sas
core_list_rule_sets.sas
core_list_rules.sas
core_update_cache.sas
core_validate_data.sas
direxist.sas
get_core_rules.sas

SAS macros

Running the CORE Engine in SAS - FCMP Functions



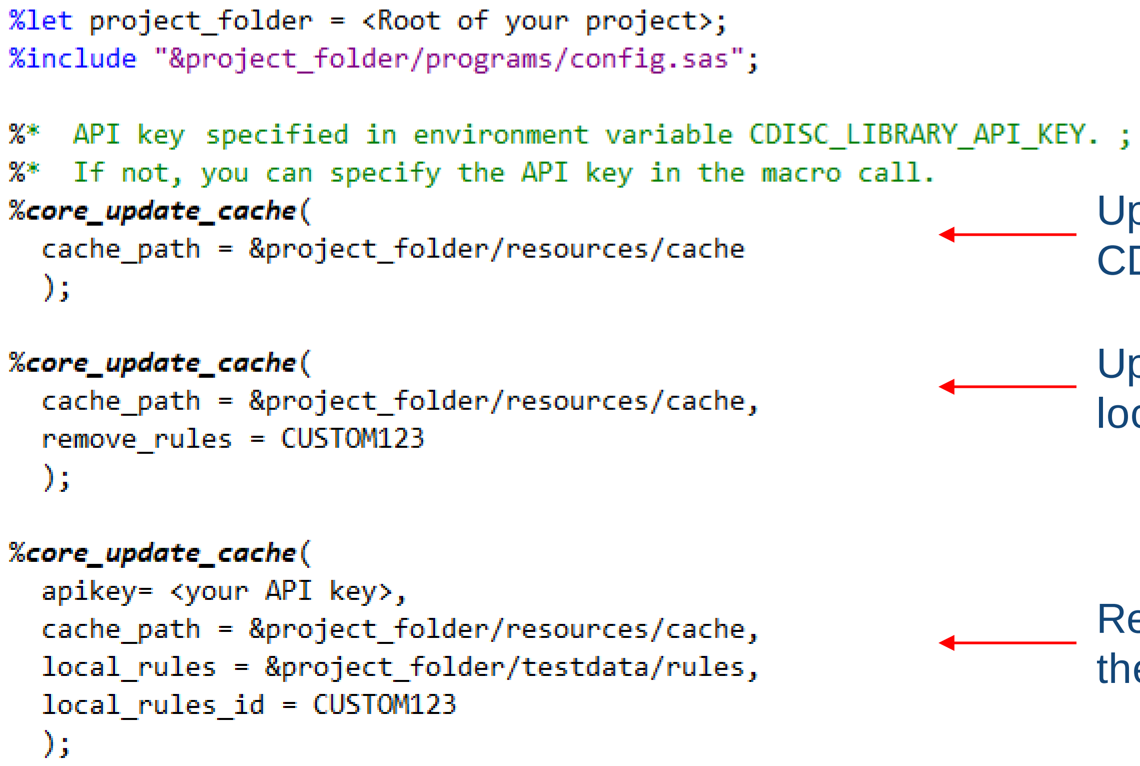
```
proc fcmp outlib = macros.core_funcs.python;

  function core_validate_data(
    cache $, pool_size, data $, dataset_path $, log_level $, report_template $,
    standard $, version $, output $, output_format $, raw_report,
    controlled_terminology_package $, define_version $, define_xml_path $,
    whodrug $, meddra $, loinc $, medrt $, rules $, local_rules $, local_rules_cache, local_rules_id) $ 128;
    length message $ 128;
    declare object py(python);
    submit into py("&project_folder/python/core_validate_data.py");
    rc = py.publish();
    rc = py.call('core_validate_data',
      cache, pool_size, data, dataset_path, log_level, report_template, standard,
      version, output, output_format, raw_report, controlled_terminology_package,
      define_version, define_xml_path, whodrug, meddra, loinc, medrt, rules,
        local_rules, local_rules_cache, local_rules_id);      You, 2 weeks ago • Uncommitted changes
    message = py.results['message_return_value'];
    return(message);
  endfunc;
```

Running the CORE Engine in SAS - Macros

```
%macro core_validate_data(
  cache_path = %sysfunc(sysget(CORE_PATH))/resources/cache,
  pool_size = 10,
  data =,
  dataset_path =,
  log_level = disabled,
  report_template = %sysfunc(sysget(CORE_PATH))/resources/templates/report-template.xlsx,
  standard =,
  version =,
  ...
  );
  ...
  %******************************************************************************;
  %* Parameter checks                                                           *;
  %******************************************************************************;
   ...
  data _null_;
    message = core_validate_data("&cache_path", &pool_size, "&data", "&dataset_path",
                "&log_level", "&report_template", "&standard", "&version", "&output",
                "&output_format",  &raw_report, "&controlled_terminology_package" ,
                "&define_version", "&define_xml_path", "&whodrug", "&meddra", "&loinc", "&medrt",
                "&rules", "&local_rules", &local_rules_cache, "&local_rules_id");
    if not missing(message) then putlog "ERR" "OR: " message;
  run;

%mend core_validate_data;
```

Running the CORE Engine in SAS

Running the CORE Engine in SAS - Update local cache



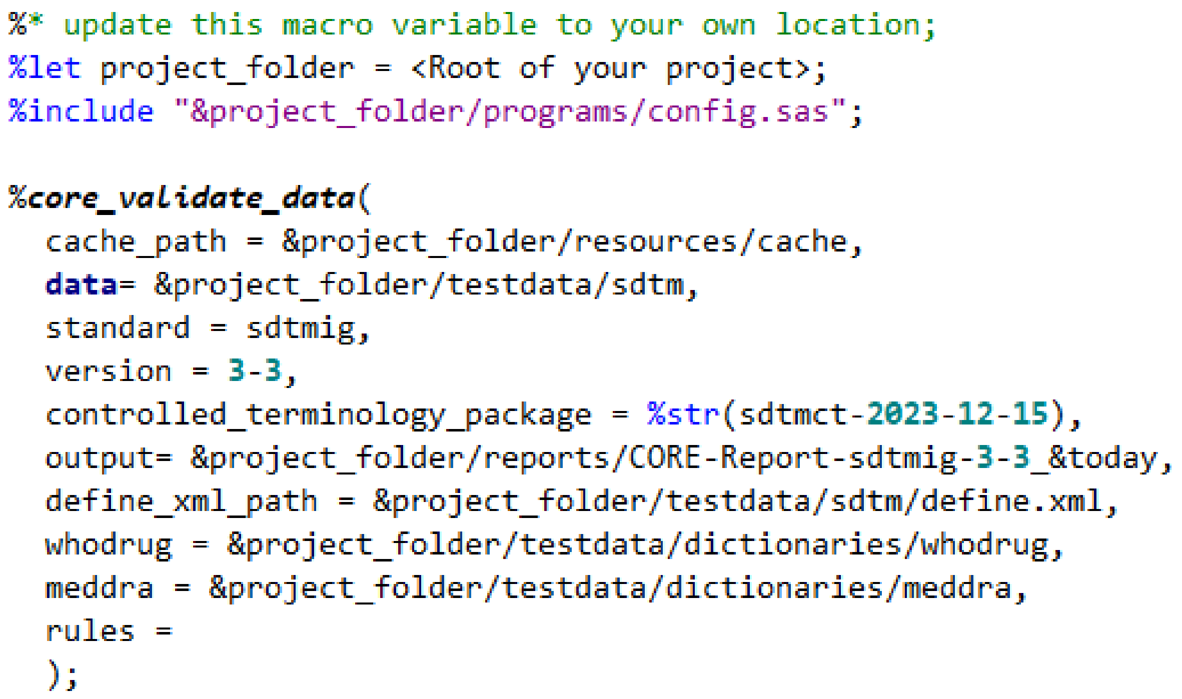
```
%let project_folder = <Root of your project>;
%include "&project_folder/programs/config.sas";

%*  API key specified in environment variable CDISC_LIBRARY_API_KEY. ;
%*  If not, you can specify the API key in the macro call.
%core_update_cache(
  cache_path = &project_folder/resources/cache
  );

%core_update_cache(
  cache_path = &project_folder/resources/cache,
  remove_rules = CUSTOM123
  );

%core_update_cache(
  apikey= <your API key>,
  cache_path = &project_folder/resources/cache,
  local_rules = &project_folder/testdata/rules,
  local_rules_id = CUSTOM123
  );
```

Update local cache with latest CDISC Library rules

Update local cache with local custom rules

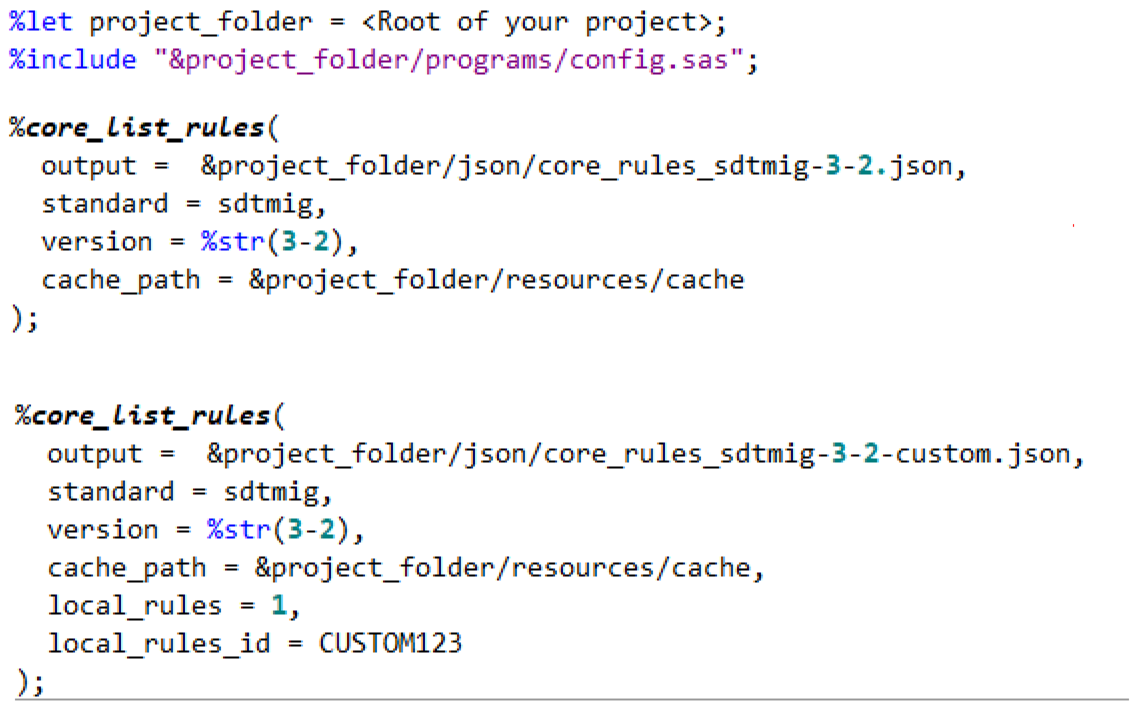
Remove custom rules from the local cache

Running the CORE Engine in SAS - Run Validation



```
%* update this macro variable to your own location;
%let project_folder = <Root of your project>;
%include "&project_folder/programs/config.sas";

%core_validate_data(
  cache_path = &project_folder/resources/cache,
  data= &project_folder/testdata/sdtm,
  standard = sdtmig,
  version = 3-3,
  controlled_terminology_package = %str(sdtmct-2023-12-15),
  output= &project_folder/reports/CORE-Report-sdtmig-3-3_&today,
  define_xml_path = &project_folder/testdata/sdtm/define.xml,
  whodrug = &project_folder/testdata/dictionaries/whodrug,
  meddra = &project_folder/testdata/dictionaries/meddra,
  rules =
  );
```

Running the CORE Engine in SAS - Get CORE Rules

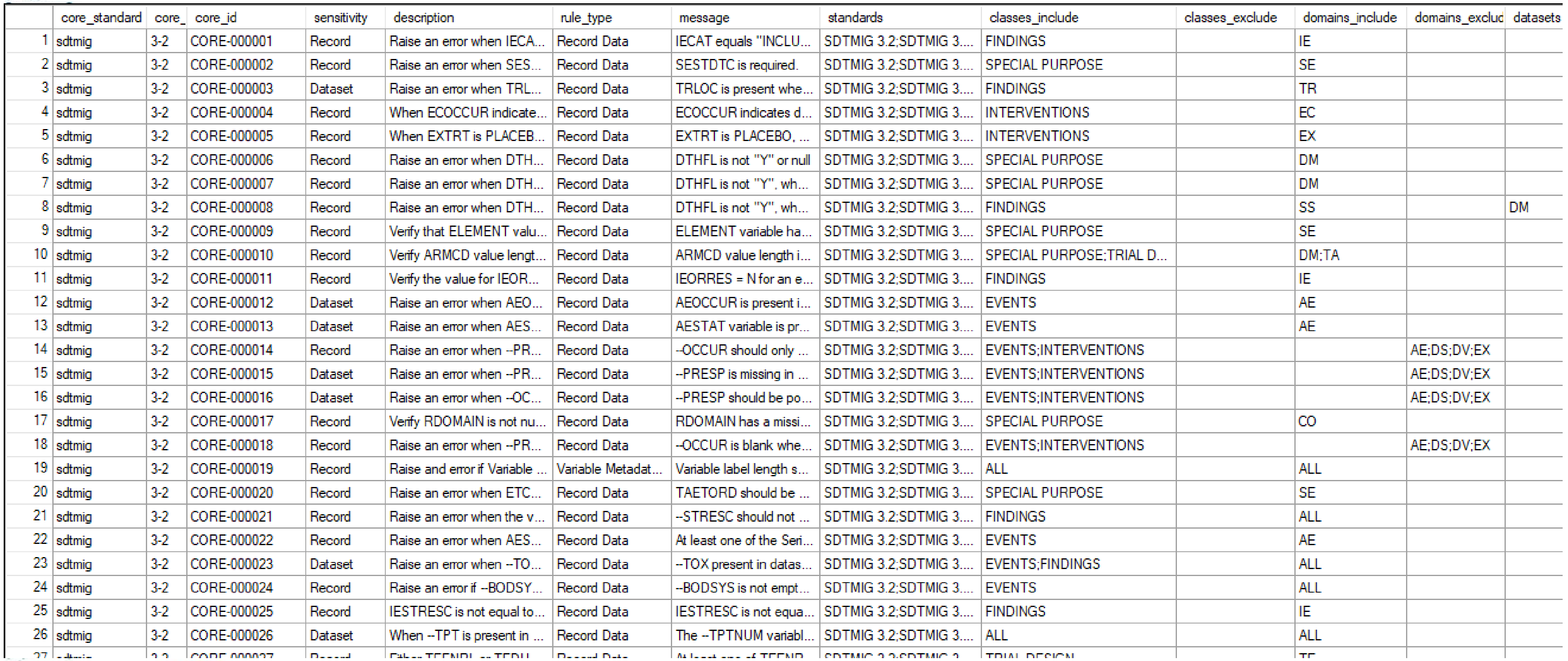


```
%let project_folder = <Root of your project>;
%include "&project_folder/programs/config.sas";

%core_list_rules(
  output =  &project_folder/json/core_rules_sdtmig-3-2.json,
  standard = sdtmig,
  version = %str(3-2),
  cache_path = &project_folder/resources/cache
);


%core_list_rules(
  output =  &project_folder/json/core_rules_sdtmig-3-2-custom.json,
  standard = sdtmig,
  version = %str(3-2),
  cache_path = &project_folder/resources/cache,
  local_rules = 1,
  local_rules_id = CUSTOM123
);
```

Get CORE rules
from local cache

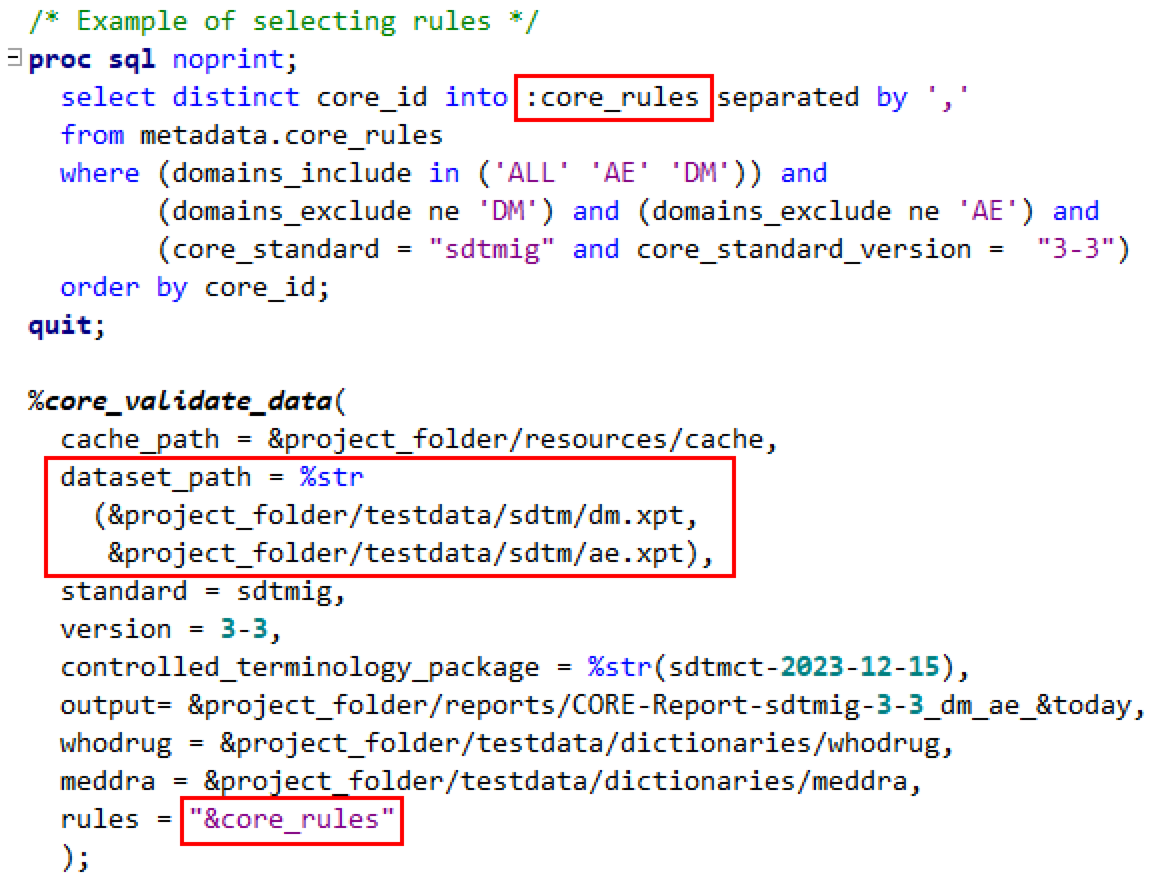
Get custom CORE rules
from local cache

Running the CORE Engine in SAS - Get CORE Rules



	core_standard	core_	core_id	sensitivity	description	rule_type	message	standards	classes_include	classes_exclude	domains_include	domains_exclud	datasets
1	sdtmig	3-2	CORE-000001	Record	Raise an error when IECA...	Record Data	IECAT equals "INCLU...	SDTMIG 3.2;SDTMIG 3....	FINDINGS		IE		
2	sdtmig	3-2	CORE-000002	Record	Raise an error when SES...	Record Data	SESTDTC is required.	SDTMIG 3.2;SDTMIG 3....	SPECIAL PURPOSE		SE		
3	sdtmig	3-2	CORE-000003	Dataset	Raise an error when TRL...	Record Data	TRLOC is present whe...	SDTMIG 3.2;SDTMIG 3....	FINDINGS		TR		
4	sdtmig	3-2	CORE-000004	Record	When ECOCCUR indicate...	Record Data	ECOCCUR indicates d...	SDTMIG 3.2;SDTMIG 3....	INTERVENTIONS		EC		
5	sdtmig	3-2	CORE-000005	Record	When EXTRT is PLACEB...	Record Data	EXTRT is PLACEBO, ...	SDTMIG 3.2;SDTMIG 3....	INTERVENTIONS		EX		
6	sdtmig	3-2	CORE-000006	Record	Raise an error when DTH...	Record Data	DTHFL is not "Y" or null	SDTMIG 3.2;SDTMIG 3....	SPECIAL PURPOSE		DM		
7	sdtmig	3-2	CORE-000007	Record	Raise an error when DTH...	Record Data	DTHFL is not "Y", wh...	SDTMIG 3.2;SDTMIG 3....	SPECIAL PURPOSE		DM		
8	sdtmig	3-2	CORE-000008	Record	Raise an error when DTH...	Record Data	DTHFL is not "Y", wh...	SDTMIG 3.2;SDTMIG 3....	FINDINGS		SS		DM
9	sdtmig	3-2	CORE-000009	Record	Verify that ELEMENT valu...	Record Data	ELEMENT variable ha...	SDTMIG 3.2;SDTMIG 3....	SPECIAL PURPOSE		SE		
10	sdtmig	3-2	CORE-000010	Record	Verify ARMCD value lengt...	Record Data	ARMCD value length i...	SDTMIG 3.2;SDTMIG 3....	SPECIAL PURPOSE;TRIAL D...		DM;TA		
11	sdtmig	3-2	CORE-000011	Record	Verify the value for IEOR...	Record Data	IEORRES = N for an e...	SDTMIG 3.2;SDTMIG 3....	FINDINGS		IE		
12	sdtmig	3-2	CORE-000012	Dataset	Raise an error when AEO...	Record Data	AEOCCUR is present i...	SDTMIG 3.2;SDTMIG 3....	EVENTS		AE		
13	sdtmig	3-2	CORE-000013	Dataset	Raise an error when AES...	Record Data	AESTAT variable is pr...	SDTMIG 3.2;SDTMIG 3....	EVENTS		AE		
14	sdtmig	3-2	CORE-000014	Record	Raise an error when --PR...	Record Data	--OCCUR should only ...	SDTMIG 3.2;SDTMIG 3....	EVENTS;INTERVENTIONS			AE;DS;DV;EX	
15	sdtmig	3-2	CORE-000015	Dataset	Raise an error when --PR...	Record Data	--PRESP is missing in ...	SDTMIG 3.2;SDTMIG 3....	EVENTS;INTERVENTIONS			AE;DS;DV;EX	
16	sdtmig	3-2	CORE-000016	Dataset	Raise an error when --OC...	Record Data	--PRESP should be po...	SDTMIG 3.2;SDTMIG 3....	EVENTS;INTERVENTIONS			AE;DS;DV;EX	
17	sdtmig	3-2	CORE-000017	Record	Verify RDOMAIN is not nu...	Record Data	RDOMAIN has a missi...	SDTMIG 3.2;SDTMIG 3....	SPECIAL PURPOSE		CO		
18	sdtmig	3-2	CORE-000018	Record	Raise an error when --PR...	Record Data	--OCCUR is blank whe...	SDTMIG 3.2;SDTMIG 3....	EVENTS;INTERVENTIONS			AE;DS;DV;EX	
19	sdtmig	3-2	CORE-000019	Record	Raise and error if Variable ...	Variable Metadat...	Variable label length s...	SDTMIG 3.2;SDTMIG 3....	ALL		ALL		
20	sdtmig	3-2	CORE-000020	Record	Raise an error when ETC...	Record Data	TAETORD should be ...	SDTMIG 3.2;SDTMIG 3....	SPECIAL PURPOSE		SE		
21	sdtmig	3-2	CORE-000021	Record	Raise an error when the v...	Record Data	--STRESC should not ...	SDTMIG 3.2;SDTMIG 3....	FINDINGS		ALL		
22	sdtmig	3-2	CORE-000022	Record	Raise an error when AES...	Record Data	At least one of the Seri...	SDTMIG 3.2;SDTMIG 3....	EVENTS		AE		
23	sdtmig	3-2	CORE-000023	Dataset	Raise an error when --TO...	Record Data	--TOX present in datas...	SDTMIG 3.2;SDTMIG 3....	EVENTS;FINDINGS		ALL		
24	sdtmig	3-2	CORE-000024	Record	Raise an error if --BODSY...	Record Data	--BODSYS is not empt...	SDTMIG 3.2;SDTMIG 3....	EVENTS		ALL		
25	sdtmig	3-2	CORE-000025	Record	IESTRESC is not equal to...	Record Data	IESTRESC is not equa...	SDTMIG 3.2;SDTMIG 3....	FINDINGS		IE		
26	sdtmig	3-2	CORE-000026	Dataset	When --TPT is present in ...	Record Data	The --TPTNUM variabl...	SDTMIG 3.2;SDTMIG 3....	ALL		ALL		
27	sdtmig	3-2	CORE-000027	Record	Either TEENRL or TEDU...	Record Data	At least one of TEENR...	SDTMIG 3.2;SDTMIG 3...	TRIAL DESIGN		TE		

Running the CORE Engine in SAS - Run Validation



```sas
/* Example of selecting rules */
proc sql noprint;
  select distinct core_id into :core_rules separated by ','
  from metadata.core_rules
  where (domains_include in ('ALL' 'AE' 'DM')) and
        (domains_exclude ne 'DM') and (domains_exclude ne 'AE') and
        (core_standard = "sdtmig" and core_standard_version =  "3-3")
  order by core_id;
quit;

%core_validate_data(
  cache_path = &project_folder/resources/cache,
  dataset_path = %str
    (&project_folder/testdata/sdtm/dm.xpt,
     &project_folder/testdata/sdtm/ae.xpt),
  standard = sdtmig,
  version = 3-3,
  controlled_terminology_package = %str(sdtmct-2023-12-15),
  output= &project_folder/reports/CORE-Report-sdtmig-3-3_dm_ae_&today,
  whodrug = &project_folder/testdata/dictionaries/whodrug,
  meddra = &project_folder/testdata/dictionaries/meddra,
  rules = "&core_rules"
  );
```

Q
U
E
S
T
I
O
N
S

ljansen@cdisc.org

lexjansen@gmail.com

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

GitHub repo: https://github.com/lexjansen/cdisc-core-sas

Open issues: https://github.com/lexjansen/cdisc-core-sas/issues

cdisc