CDISC CORE for Conformance Checking in Data Automation Pipelines

Sam Hume, DSc, CDISC

PHUSE SDE Collegeville, PA

2024-08-29

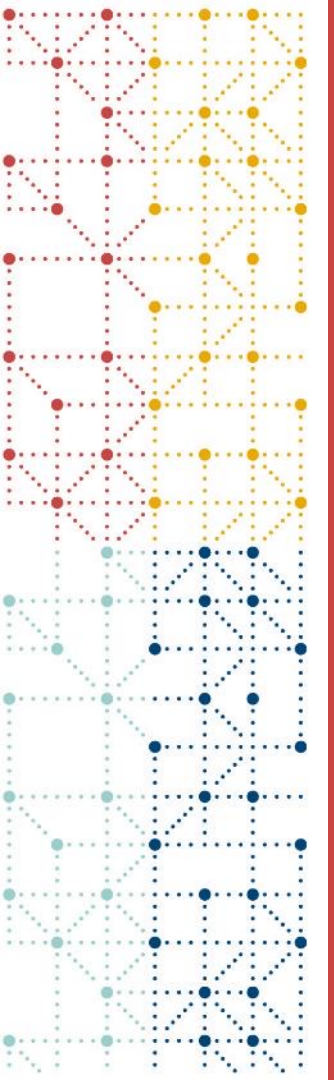
Meet the Speaker

Sam Hume

Title: Principal Consultant

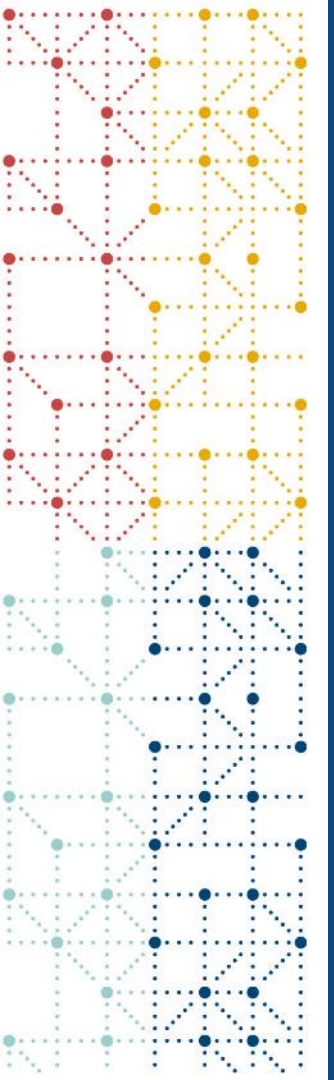
Organization: CDISC

Sam Hume co-leads the CDISC Data Exchange Standards team, advises CDISC leadership on strategy and technical topics, and contributes to COSA, CORE, CDISC Library, and other CDISC projects. Sam formerly served as the CDISC VP of Data Science. During his 30 years in the biopharmaceutical industry, he has held several senior-level technology positions. Sam is an active PHUSE contributor. He holds a doctorate in Information Systems.



Agenda

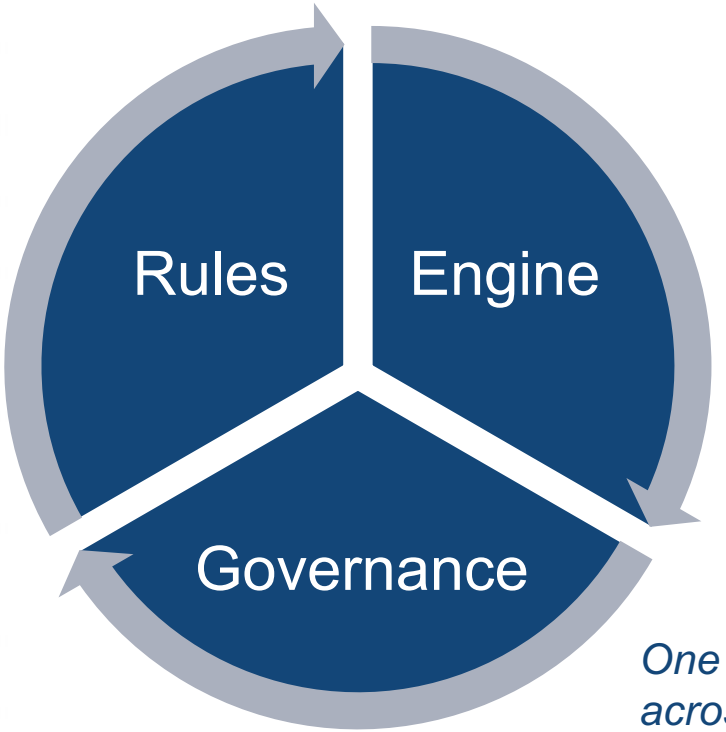
1. CORE Overview
2. Conformance Rules
3. Extending CORE
4. Running the CORE Engine
5. Using the CORE Engine
6. What’s next



Overview

A bit of background information to get us started

What is CORE?



- **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

One set of aligned and transparent conformance rules used across regulatory, sponsors, and vendors along with central curation and governance of the rules

CORE Software: Engine and Rule Editor

- Each project
  - Has a public GitHub repository on the cdisc-org account and is listed on the COSA Directory
  - Has been released under the MIT open-source license
  - Development is led by CDISC
  - Still under development, but are being actively used
  - Can be extended (supports the development of software extensions)

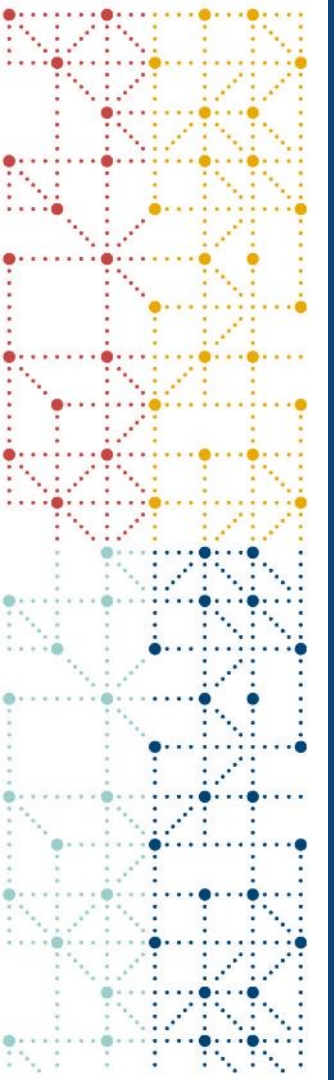
- CORE Engine
  - Written in Python
  - Makes use of the Venmo Business Rule Engine

- CORE Rule Editor
  - Written in TypeScript
  - Makes use of the VSCode editor

cdisc

Key Idea: CORE will be deployed in multiple ways

- Anticipate many will deploy CORE in multiple ways
  - Using CORE within a vendor's platform (e.g., SAS)
  - Setting up CORE to run in your organization's cloud environment (i.e., CORE Service)
  - Running a desktop version of CORE (i.e., manually initiated)
  - Running the command-line version of CORE (i.e., used in automation pipelines)
  - Running CORE from a docker container (i.e., used in automation pipelines)
  - Running the CORE rules using alternative engines
  - Building tools that incorporate the CORE Engine
- CORE can be run at no cost, allowing organizations to run the CORE rules using a mix of deployment options
- CORE rules may be used in conjunction with other rule engines
- CORE rules may be developed for additional scenarios beyond submissions



Conformance Rules

Creating your own data quality and conformance rules is part of using CORE in a data automation pipeline

EDIT TEST DIFF

```yaml
# Variable: EXMETHOD
# Condition:
# Rule: EXMETHOD not present in dataset
Authorities:
  - Organization: CDISC
    Standards:
      - Name: SDTMIG
        References:
          - Citations:
              - Cited Guidance: Method of administration of the treatment. Not to be used with
                  human clinical trials.
                Document: Model v1.7
                Item: EXMETHOD
                Section: Table 2.2.12.1
            Origin: SDTM and SDTMIG Conformance Rules
            Rule Identifier:
              Id: CG0568
              Version: '1'
            Version: '2.0'
        Version: '3.3'
Check:
  all:
    - name: EXMETHOD
      operator: exists
Core:
  Id: CORE-000326
  Status: Published
  Version: '1'
Description: Trigger error when EXMETHOD exists in the EX dataset for human clinical trials
Executability: Fully Executable
Outcome:
  Message: EXMETHOD is not to be used with human clinical trials
Rule Type: Record Data
Scope:
  Classes:
    Include:
      - INTERVENTIONS
  Domains:
    Include:
      - EX
Sensitivity: Record
```

CORE Rule Editor

Web-based application, no software to install

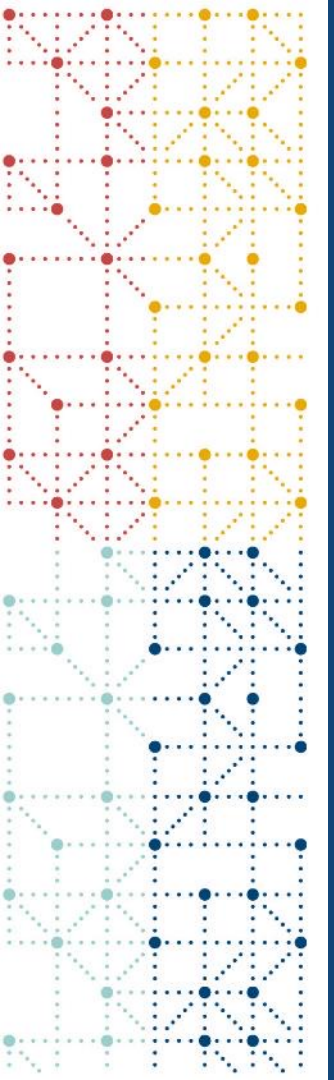
- Structured document, 1 CORE rule per file containing rule's metadata & check logic
- Real-time syntax checking
- Can be installed in your environment: cloud or local server
- You can develop and maintain your own rules
- Rule Editor Reference Guide

CDISC Non-foundational Standards

- Research Collaboration Agreement with US FDA CDER & CBER
  - CDISC and FDA working together to develop executable formats of the FDA Business Rules
- TransCelerate Digital Data Flow
  - Conformance Rule Proof of Concept on Unified Study Definition Model (USDM)
  - 100 rules defined and partially created
- Tobacco Implementation Guide
  - Creating new TIG-specific rules
  - 1240 rule defined
- Define-XML and JSON schema checking
- Sponsor-defined, custom rules for conformance and data quality

Key Idea: Not limited to CDISC rules and datasets

- For use in a data pipeline, create your own targeted rule sets
  - Might include your own rules
  - Test for conformance and data quality
  - Use for DTA checks
- For use outside of the traditional CDISC datasets, such as SDTM or ADaM
  - Used to check USDM content – not a tabular dataset
  - Used to check Define-XML
  - Used to check tobacco studies
  - Has been used for real-time data checking



Extending CORE

How to extend the CORE Engine

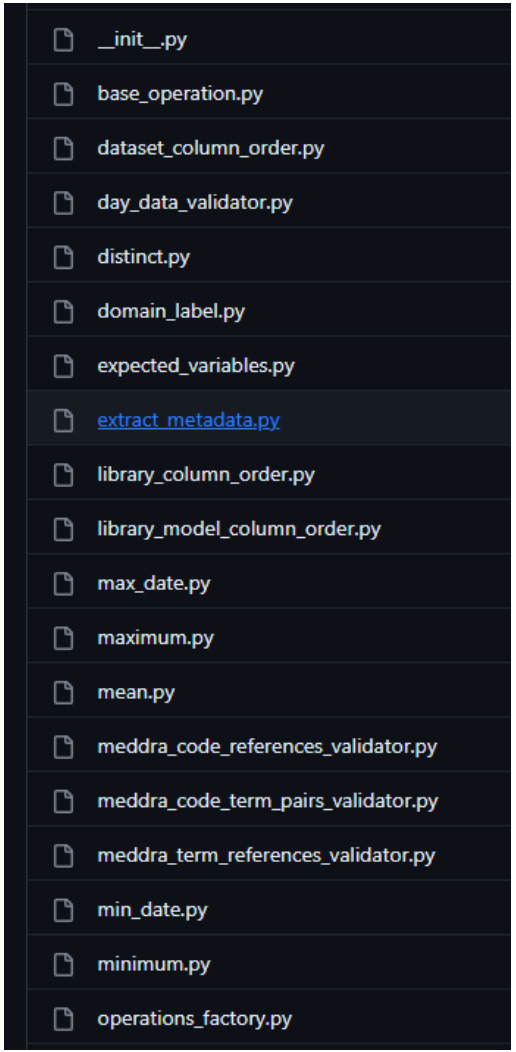
CORE Engine extensibility

- Operations
  - Define an operation on a dataset, e.g., variable_permissibility, mean
- Dataset Builder
  - Used to define a dataset to match a rule type
- Dataset Reader
  - Used to define dataset formats for reading, e.g., SAS v5 XPORT, Dataset-JSON, CSV
- Data Service
  - Define the service from which the dataset will be read, e.g., local, Azure, AWS
- Checks
  - Used in rule tests, e.g., equal_to, non_empty, matches_regex
- Cache
  - Used to interface with a cache for rules and metadata, e.g., in memory, Redis
- Reporting
  - Defines a type of reporting, e.g., Excel, JSON
- Logging
  - Specifies what and to what level of detail logs are generated

Extending CORE: Adding an Operation

Operations:

- Typically used to pre-process data to facilitate the use of Checks
- May generate new dataset columns with values that can be referenced in a rule
- Example operations:
  - distinct
  - max_date
  - mean
  - variable_exists
  - variable_permissibility
  - Many more…
- Easily add new operations



Creating a new operation

- Inherit the Base Operation and implement the _execute_operation method

```
from cdisc_rules_engine.operations.base_operation import BaseOperation
from typing import List


class IsOdd(BaseOperation):
    def _execute_operation(self):
        """
        Returns True if the target variable is odd, else return false
        """
        return self.params.dataframe[self.params.target] % 2 != 0
```

- Register the method so the engine can use it
- Update the rule schema
- Implement a rule that uses the operation

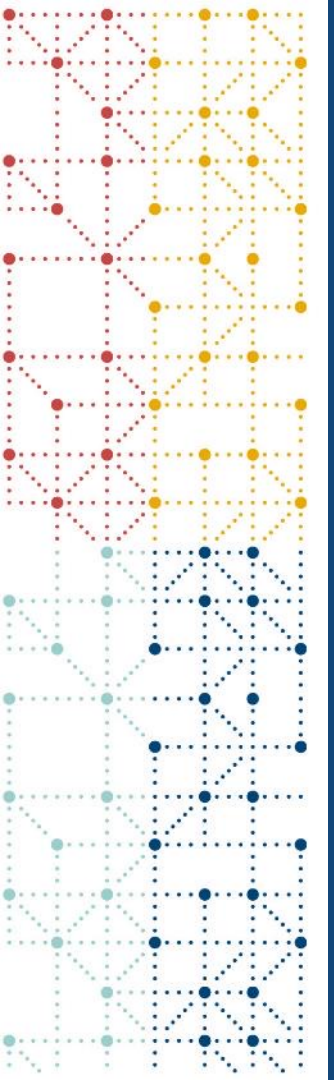
Create a rule that uses the new is_odd operation

- The is_odd operation is used to create a new column that contains “true” if AGE is an odd number
- The Check examines “all” records to find cases where the $age_is_odd column equals “true”
- A report is generated identifying cases where this rule fired
- This could have been implemented as a check operator instead of an operation

```
all:
  - name: $age_is_odd
    operator: equal_to
    value: true
Operations:
  - id: $age_is_odd
    name: AGE
    operator: is_odd
```

Key Idea: extensibility and adaptability

- Everything that CORE does can be extended to address novel needs
  - CORE has been adapted to test non-tabular data for DDF USDM
  - CORE has been used to develop organization-specific rules
  - CORE has been run on platforms for which there was no release package
  - CORE has been run locally, on on-prem servers, and in the cloud
  - CORE runs against SAS V5 XPORT but also supports Dataset-JSON
- You can write rules that incorporate new kinds of testing
- Implementers develop tools to extend CORE
  - Conformance results visualizations and dashboards
  - UI for manually executing rules



Running the CORE Engine

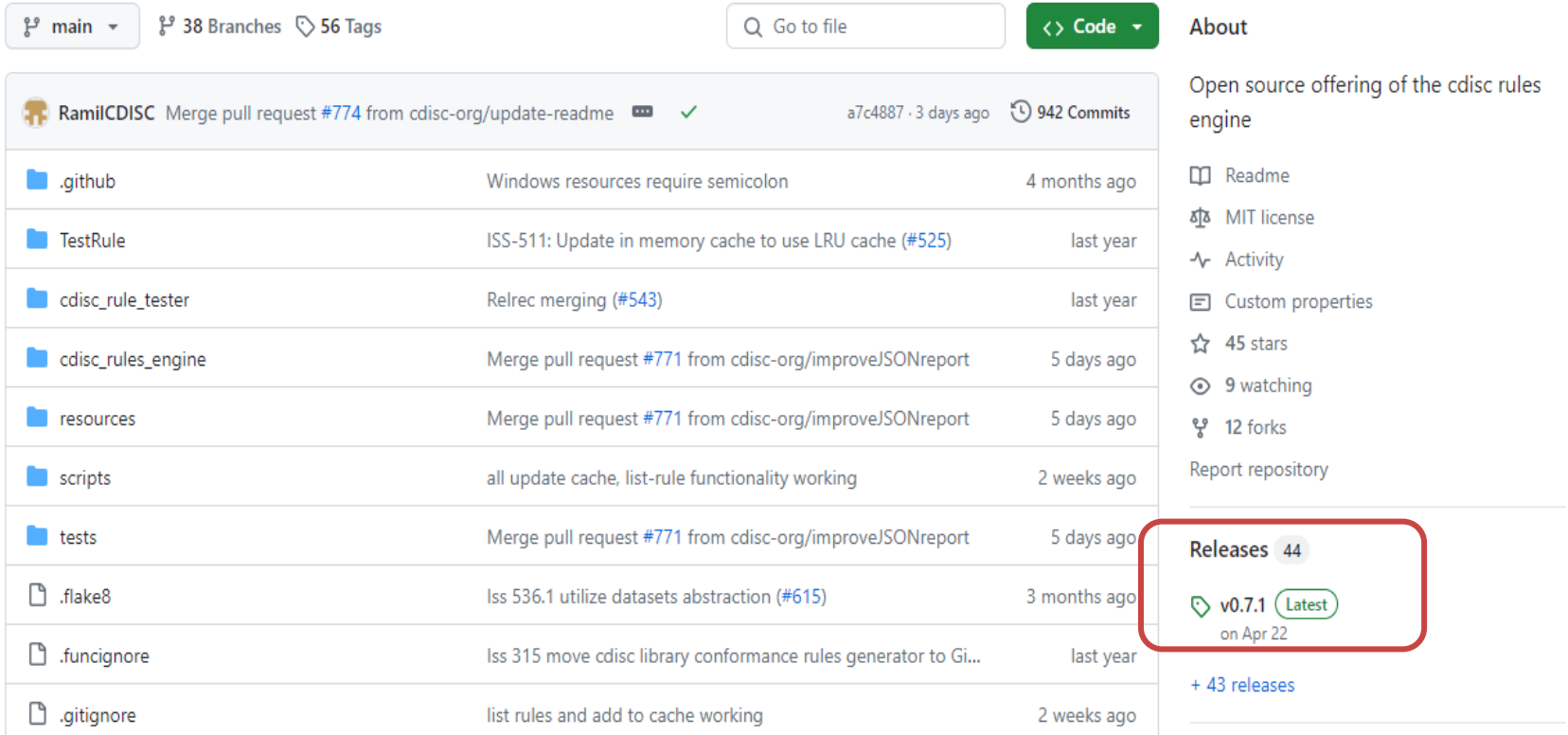
How to run the CORE Engine today

Running the CORE Engine

- CLI executable available in GitHub
  - Cached rules
  - Windows, Mac, and Linux install packages
  - Unzip and run – will need datasets to test
- Engine available on PyPI
  - Engine is a component that can be used in your own code
- Python source code
  - https://github.com/cdisc-org/cdisc-rules-engine?tab=readme-ov-file#running-a-validation
  - Clone the repository and run the Python source code
- CORE container (docker)
- Desktop versions
  - Vendor-released versions of CORE that include a user-friendly UI

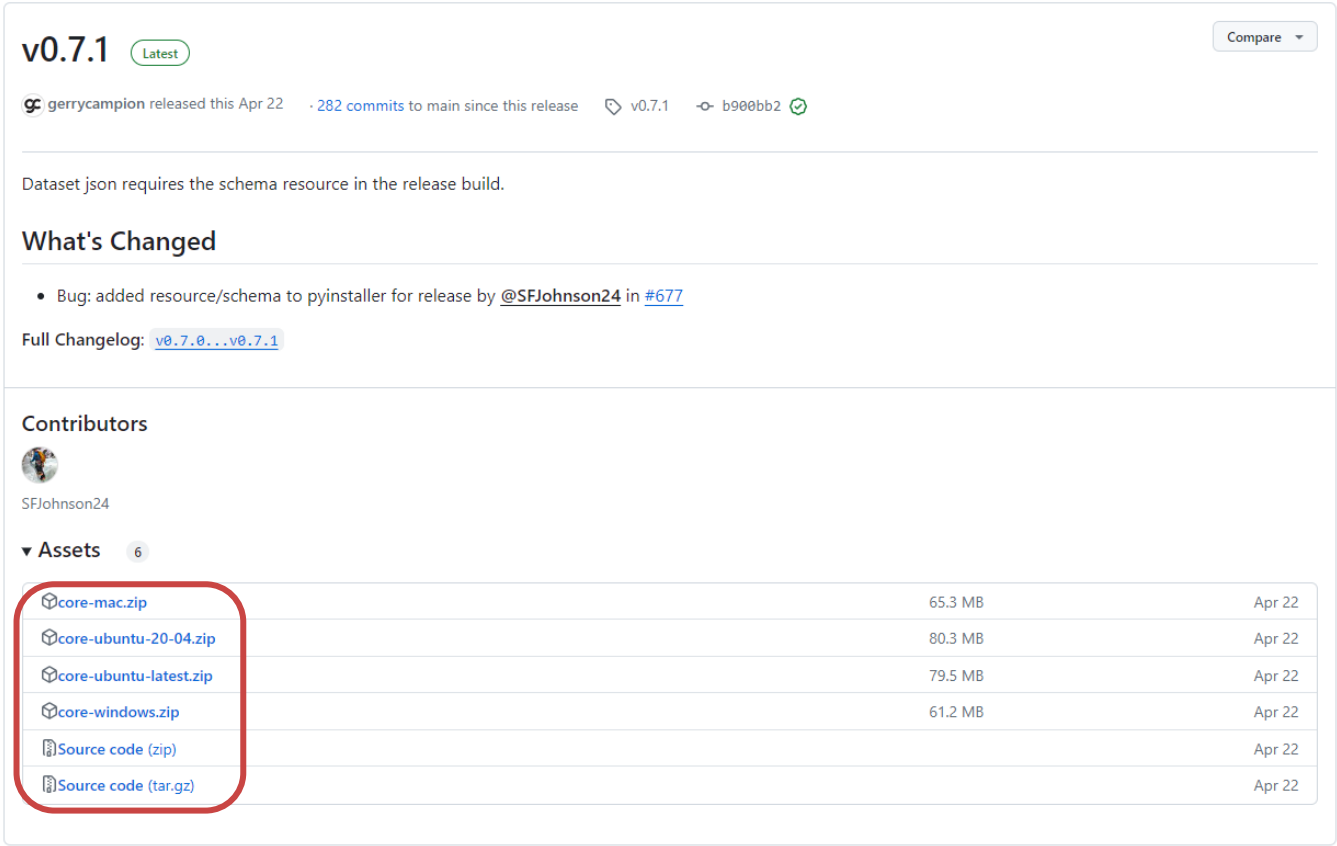
https://github.com/cdisc-org/cdisc-rules-engine

CLI Deployment – in GitHub under Releases



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

Download the latest CLI CORE Engine



v0.7.1 Latest

Compare

gerrycampion released this Apr 22 · 282 commits to main since this release v0.7.1 b900bb2

Dataset json requires the schema resource in the release build.

What's Changed

- Bug: added resource/schema to pyinstaller for release by @SFJohnson24 in #677

Full Changelog: v0.7.0...v0.7.1

Contributors

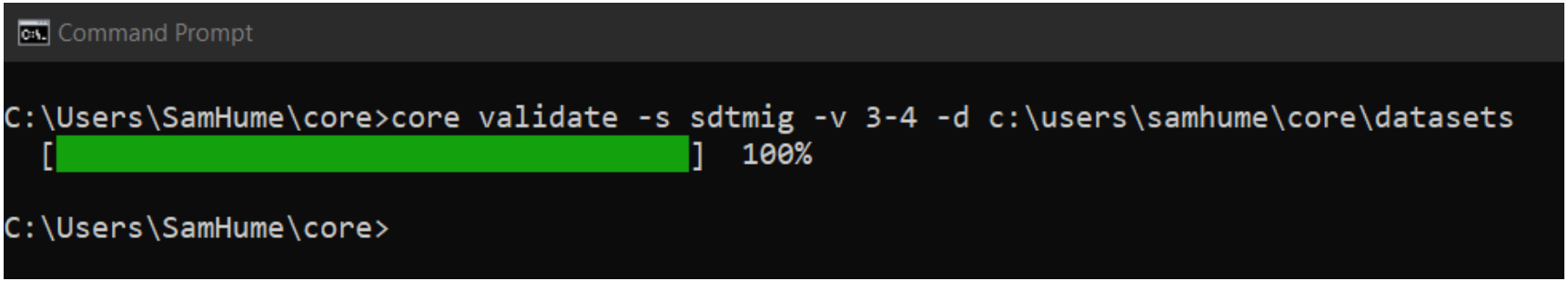
SFJohnson24

▾ Assets 6

core-mac.zip	65.3 MB	Apr 22
core-ubuntu-20-04.zip	80.3 MB	Apr 22
core-ubuntu-latest.zip	79.5 MB	Apr 22
core-windows.zip	61.2 MB	Apr 22
Source code (zip)		Apr 22
Source code (tar.gz)		Apr 22

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

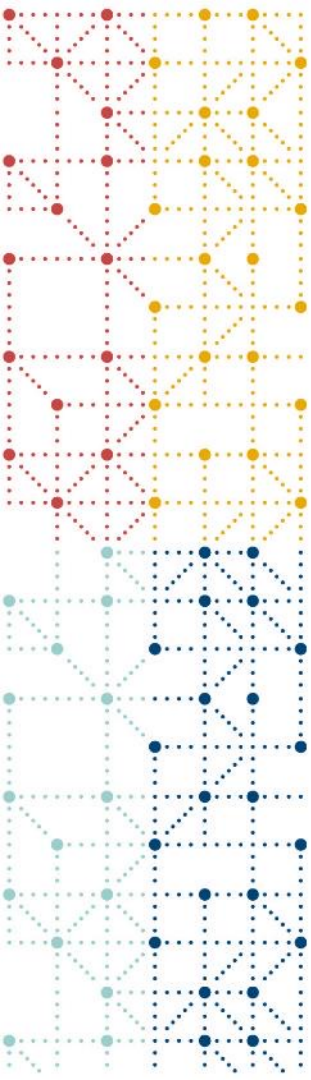
Running CORE at the command-line



- Above shows running the CORE Engine on Windows
- Used SDTMIG v3.2 test data (with optional Define-XML file)
- See README.md documentation in the GitHub repository
  - c:\>core --help

Key Idea: there are multiple ways to implement CORE

- CORE can be downloaded as a ready-to-run executable
- CORE may be embedded in another system
- CORE can be implemented as a library within your Python code
- CORE can be accessed and run using the existing source code
- CORE from the command-line is ideal for data automation pipelines



Using the CORE Engine

Selected examples where CORE has been used in data automation pipelines.

Regulators & CDISC Partners

RCA with FDA CDER/CBER to develop executable FDA business rules

BUSINESS WIRE

CDISC is Proud to Announce a Research Collaboration to Incorporate FDA Business Rules into CDISC's Open Rules Engine (CORE)

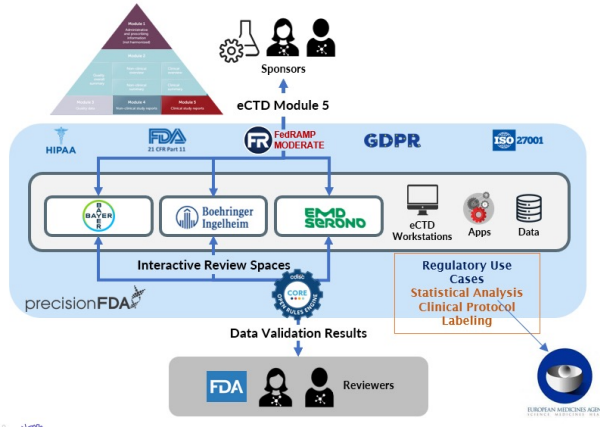
Provided by Business Wire
Jan 16, 2024 12:32 PM EST

CDISC is Proud to Announce a Research Collaboration to Incorporate FDA Business Rules into CDISC's Open Rules Engine (CORE)

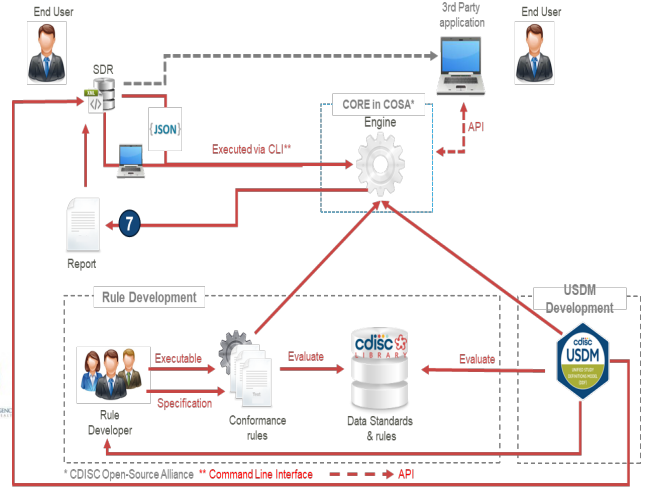
CDISC is proud to announce a research collaboration with the U.S. Food and Drug Administration's Office of Translational Sciences in the Center for Drug Evaluation and Research and Office of Regulatory Operations in the Center for Biologics Evaluation and Research to incorporate FDA Business Rules into CDISC's Open Rules Engine (CORE).

CDISC's CORE project provides an open-source version of the CDISC Conformance Rules in a machine-executable format. These rules, published and managed by CDISC, create a single source for conformance rules and allow external vendors and sponsor companies to implement and extend these rules within their tools. FDA Business Rules are currently written in a plain

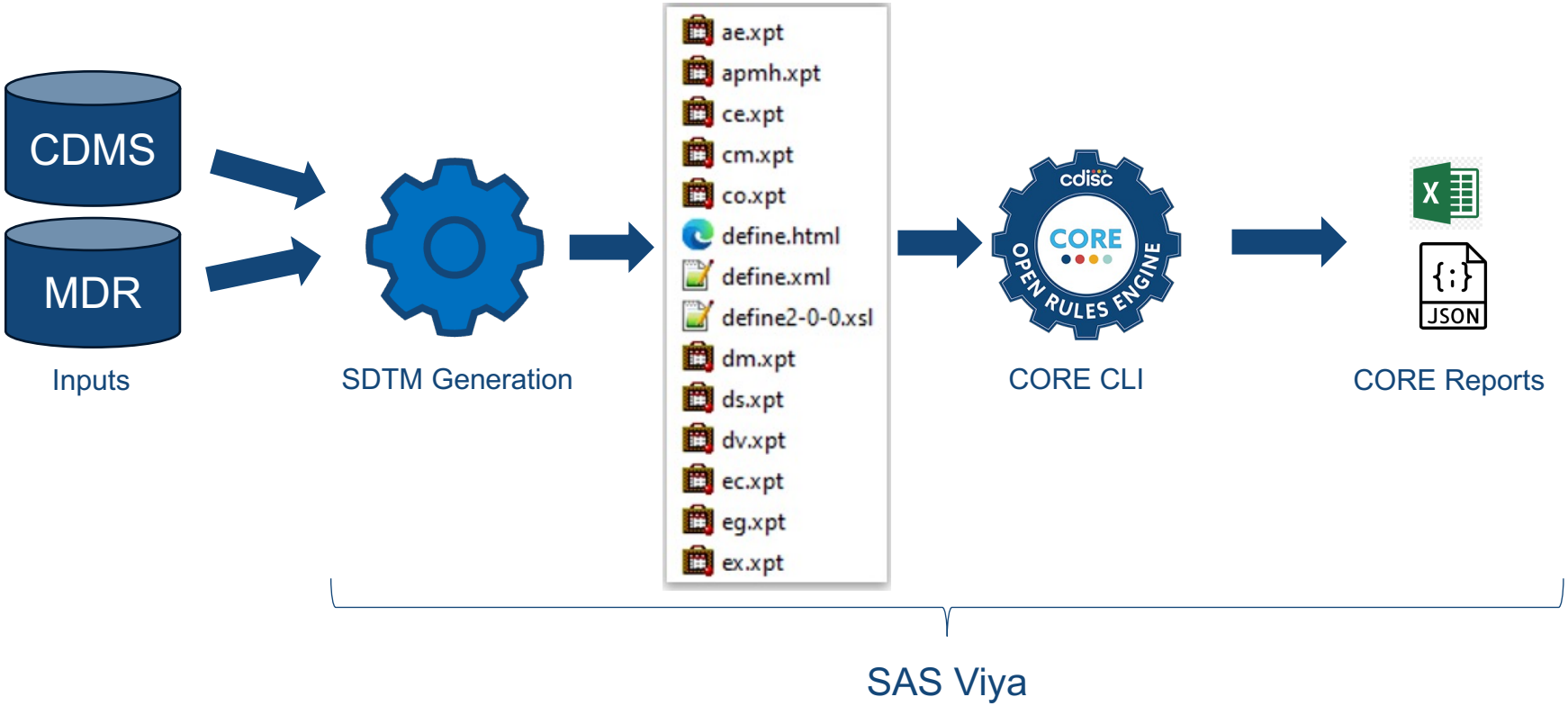
POC with FDA PRISM to support TCG rejection criteria and other use cases



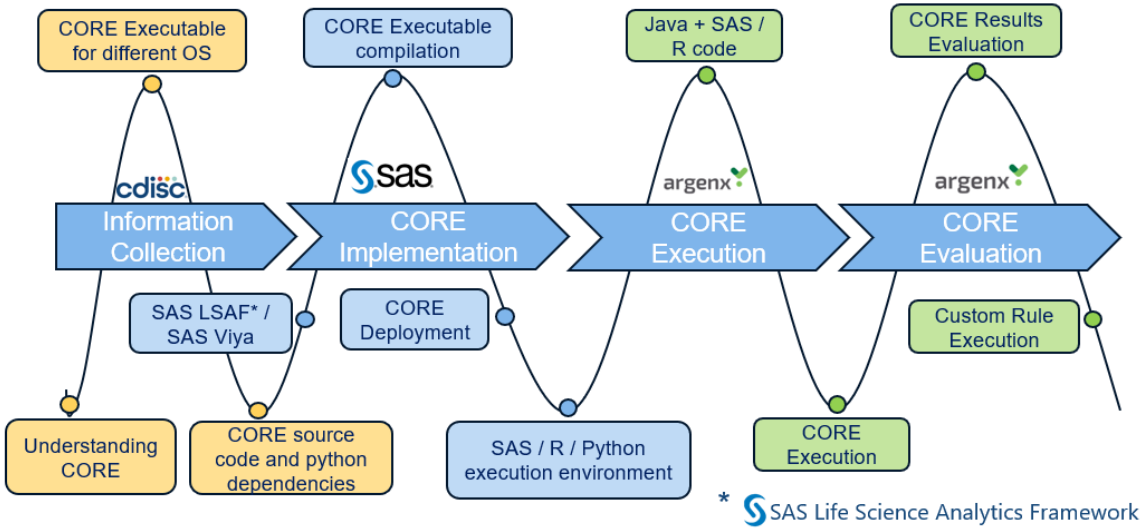
Pilot to build USDM/M11 Digital Protocol Conformance Rules



Novo Nordisk: Automated Validation



Argenx: Implementing CDISC CORE in a cloud-native, regulated environment



Start-up Challenges

- **Different Operating System**
  - CDISC executable compiled for windows and Linux (ubuntu) whereas SAS LSAF uses CentOS Linux.
  - CDISC source code needed to be compiled into executable for SAS LSAF CentOS by downloading all dependencies.
- **Execution Permission**
  - CDISC Core executable required execution permission which required working with SAS team.
  - SAS LSAF does not allow command line execution using Base/SAS
- **Programming Language(s) support**
  - Additional source code required to be written in Java, compiled into executable jar and wrapped into SAS macro to support Core execution via SAS
  - Source code written in R programming language too, to execute Core in SAS LSAF.
- **Shared Workspace**
  - Multiple users can access same executable

cdisc

SGS: Our Road to Adopting CORE: An Implementation Journey on SDTM Dataset Validation

#	CORE Benefits
1.	Rules that generate simple output Conformance = no output = no work
2.	Easy creation of rules, low code, efficient workflow
3.	Automatic upload of new SDTM or regulatory conformance rules
4.	Split up rules in validation purpose = clear categories
5.	Option to validate incoming vendor file and outgoing SDTM converted dataset

#	CORE Meets Requirements
1.	Create additional SDTM conformance rules
2.	DTA compliance
3.	Data content
4.	Other data structures

Lindus Health: How we Achieved Real-Time Validation by Integrating CORE into our Django (Python) Clinical Trial Platform

Our goal is to bring validation as close to collection as possible:

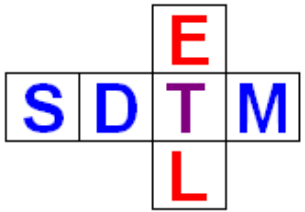
- Prevent the creation of invalid records
- Integrate CORE into our CRFs
- In-product summary of all issues
  - Locating problematic records in one click
- Avoid the use of SAS XPT files

Barriers: Speed, Depth of Integration, User Experience, Need More Dirty Data for Testing

Used CORE PyPI library to integrate data quality and conformance checks

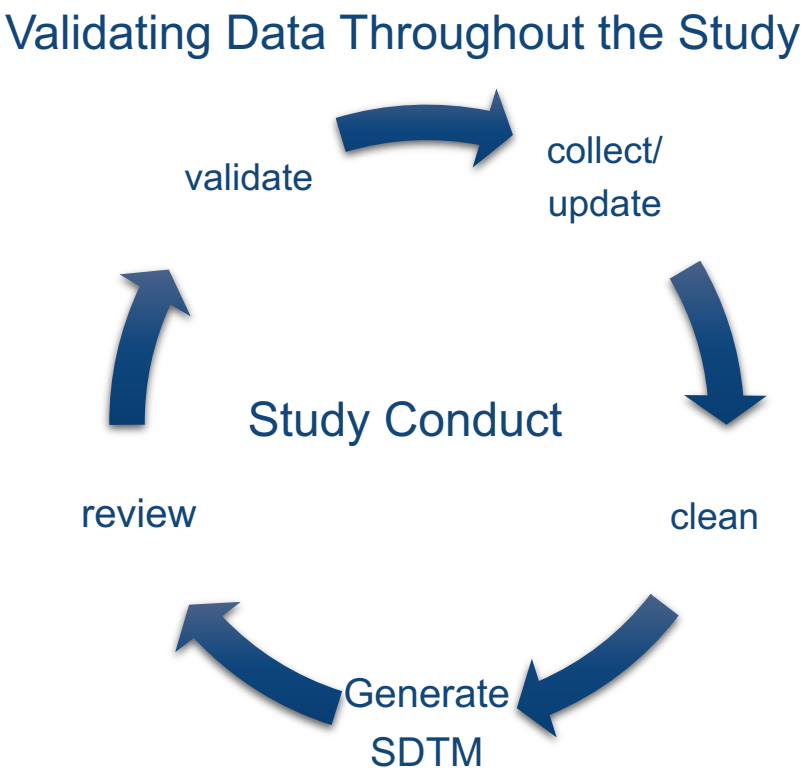
XML4Pharma: CORE Implementation in Software SDTM-ETL, a "smart mapping tool"

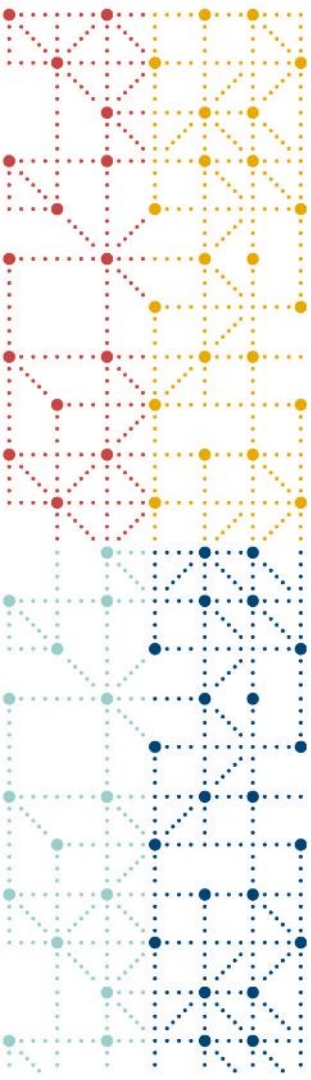
- When generating SDTM/SEND datasets, one immediately wants to validate against the standard
  - with the possibility to use a subset of rules only
  - with the possibility to select a set of SDTM/SEND files rather than all
  - In an extremely user-friendly way ...
- CDISC CORE is easy to implement in software - trigger a separate process
- As all rules and their implementations are open source and publicly available
- Not all rules have been implemented yet (but steadily growing)



Key Idea: validation throughout the study

1. CORE is being used in data automation pipelines today
2. Today, CORE primarily complements other conformance-checking tools
3. CORE users can create their own rules or rule sets
4. All CORE features are accessible via the command line
5. The CORE Rule Editor can be run in your environment
6. More work to do on CORE





What’s Next

Some insights into what’s coming next

CORE Registered Solution Provider

- Program purpose
  - For CORE vendors (solution providers)
    - Certify with CDISC that their solutions correctly use the Conformance Rules
  - For CDISC
    - Treat all CORE vendors equally
    - Achieve a level playing field regarding use of any Engine with the Conformance Rules
    - Inform the community which solutions have been certified

- Testing for certification will include
  - Generating results with Conformance Rules and test study data reflecting an “average study”
  - No system functionality testing

New CORE Engine Features

- Recent enhancements:
  - New ability to customize checks to improve overall data quality
  - Enhanced deployment flexibility using containerization
  - New cache functionality that enhances performance and flexibility in running checks
  - External dictionaries
  - Continued improvement to add check functionalities
  - The next release is scheduled before CDISC US Interchange

- High-impact items coming soon:
  - Large datasets and performance optimizations
  - Sponsor-defined, locally deployed rules
  - Container-based deployment to make it easier to begin using
    - Possibly add Kubernetes to our Docker container deployments

Key Idea: seeking to accelerate CORE development

1. **CDISC Open Rules Accelerator Program** seeking participants
2. CORE Certification Program makes it easier for organizations to use multiple engines to run the same rules
3. New features that improve performance and scalability

Thank You!

Questions?

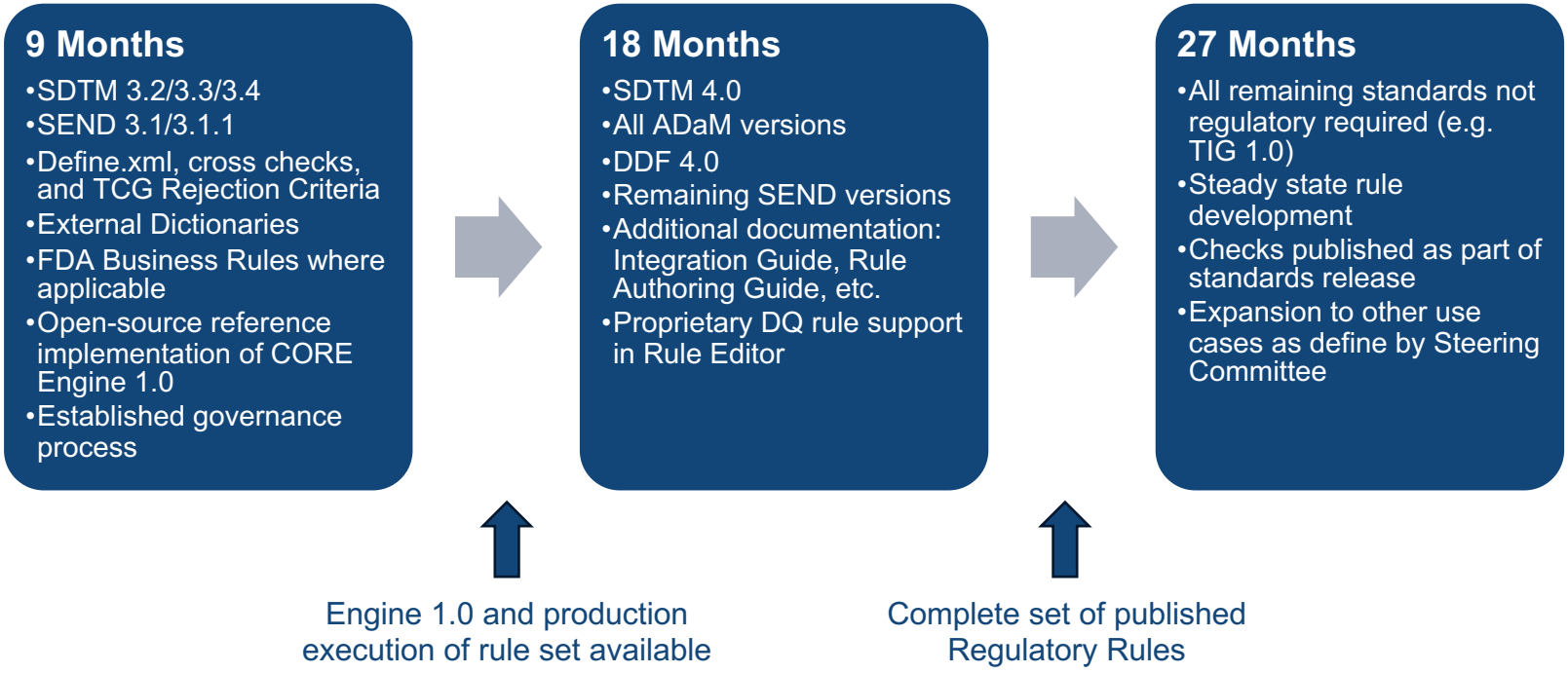
shume@cdisc.org

https://www.linkedin.com/in/sam-hume-dsc

Supporting Slides

Supporting information that will not be covered directly during the presentation.

CORE Proposed Roadmap*



* Assuming required industry resource and funding commitment