Paper DS08

Case Study: Leveraging CDISC CORE for Proactive SDTM Compliance and Submission Readiness

Shivani Gupta, Clymb Clinical, Burlington, MA, United States

ABSTRACT

Ensuring compliance with CDISC standards is essential in preparing clinical trial datasets for regulatory submission. Traditionally, validation using commercial tools is performed only after all Study Data Tabulation Model (SDTM) domains are developed, often leading to late discovery of issues and costly rework.

This paper describes our experience integrating the CDISC Open Rules Engine (CORE), an open-source validation tool, directly into the SAS environment to run compliance checks during SDTM dataset development. By embedding CORE into the workflow, we identified and resolved conformance issues earlier, improved data quality, and streamlined the overall submission process.

We also compare outcomes of per-domain validation versus study-level validation, demonstrating how early checks reduce errors, accelerate timelines, and encourage a compliance-first culture. Finally, we highlight CORE's advantages over commercial tools - including transparency, flexibility, and cost-effectiveness - thereby positioning this proactive strategy as a scalable solution for clinical programming teams to enhance efficiency and ensure regulatory readiness.

INTRODUCTION

ABOUT CDISC CORE

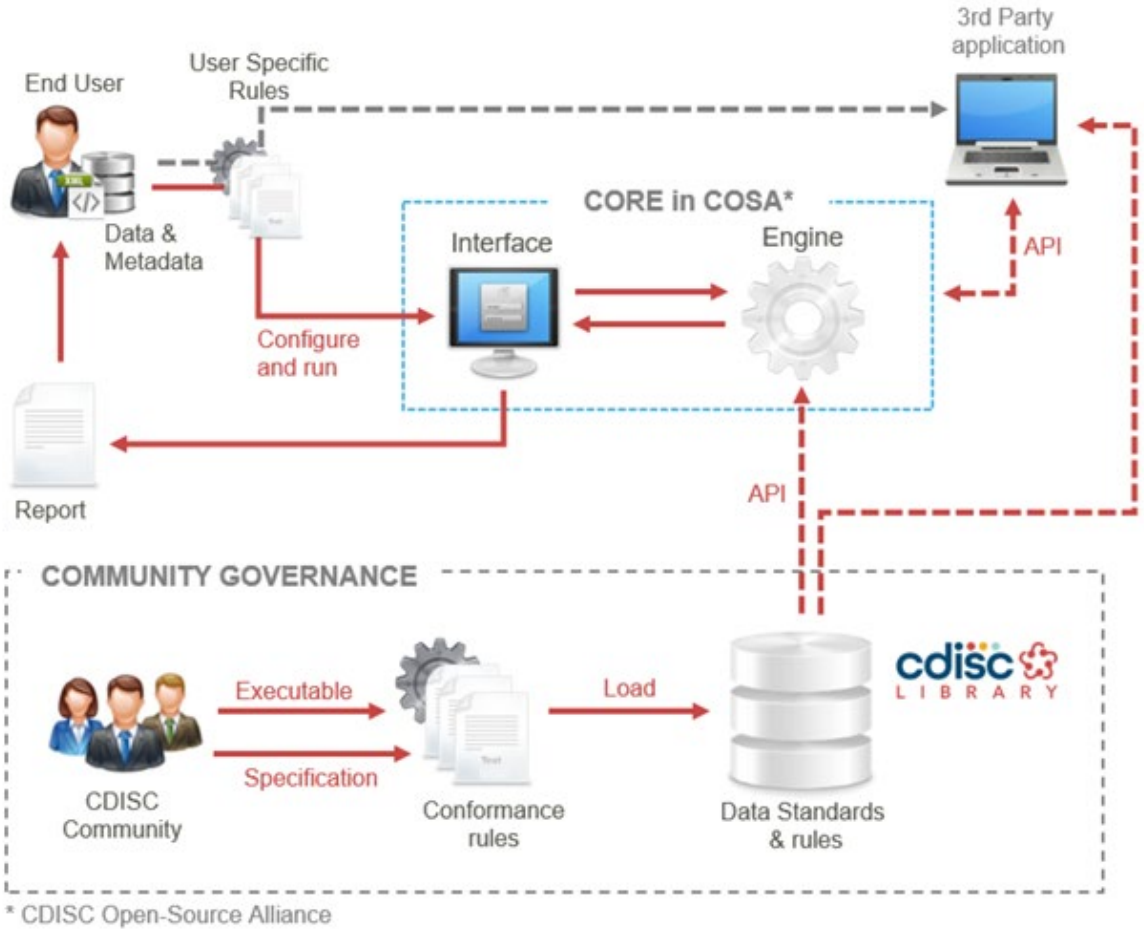
The CDISC Open Rules Engine (CORE) [1] is an open-source framework that enables consistent, transparent, and reproducible execution of CDISC conformance rules. Rather than functioning as a standalone validation tool, CORE defines a standardized engine specification that allows conformance rules to be executed across multiple platforms and implementations. This design supports both CDISC-published rules and user-defined rules, enabling flexible integration into existing clinical data workflows.

By leveraging machine-readable rule definitions, CORE standardizes conformance checking, reduces manual validation effort, and facilitates automation of clinical data quality processes. As part of the CDISC Open-Source Alliance (COSA) [2], CORE plays a key role in modernizing clinical data validation and promoting standards-based quality assurance practices across the industry.

CORE consists of two primary components: the Conformance Rules and the Conformance Rules Engine.

1. The Conformance Rules are developed and governed by CDISC, with input from the broader CDISC Community, following the same formal process used for all CDISC standards. These rules are maintained in the CDISC Library alongside other CDISC standards artifacts.

2. The Conformance Rules Engine is an open-source software application designed to apply these rules to clinical datasets and generate validation results. The Engine is publicly available on GitHub and may be deployed across a range of environments, including cloud platforms, on-premises servers, and desktop systems [3]. During execution, the Engine retrieves rules directly from the CDISC Library via the Library API, while also allowing users to define and execute custom rules. The Engine is implemented in Python and released under a permissive MIT license, supporting broad adoption and flexible integration.



IMPLEMENTATION OF CDISC CORE IN A BASE SAS® ENVIRONMENT

Method

The objective of this work was to implement the CDISC Open Rules Engine (CORE) within a BASE SAS® environment that supports restricted execution environments and ensures reproducibility across studies. The implementation followed the architecture and guidance described by Jansen in the PharmaSUG 2025 paper *"Running the CDISC Open Rules Engine (CORE) in BASE SAS®"* (SD-044) [4], which demonstrates how CORE command-line functionality can be exposed to SAS through Python integration rather than direct operating system calls.

The implementation process consisted of the following steps:

- Cloning the GitHub repository cdisc-core-sas, which accompanies the PharmaSUG SD-044 paper.
- Make sure that Python is installed in the PC.
- Follow the installation steps from the paper (create virtual environment, install their dependencies, create the system environment variables).
- Update provided macro call from program *create_core_functions.sas* based on the paper's instruction and run it once.
- Update program *config.sas* based on the paper's instruction. Include this program in all other programs from subsequent steps.
- Create the program containing all the supporting macro calls *(core_update_cache, core_list_ct, core_list_rule_sets, core_list_rules* based on the paper's instruction. Name of the program: *core_update_all.sas*.

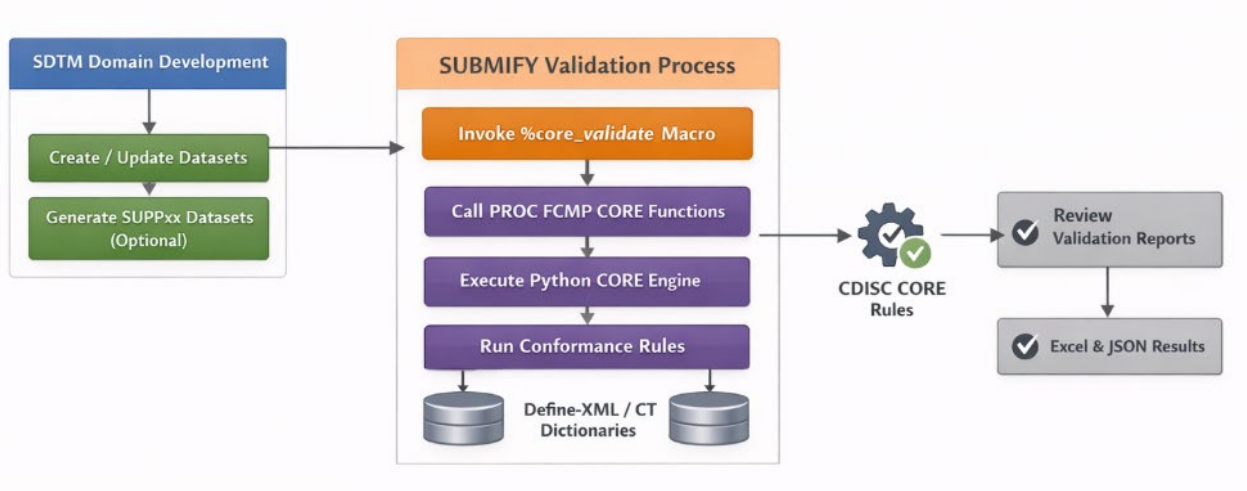
- Set up the automatization to run *core_update_all.sas* once a week to get the latest sets of rules and update the solution metadata.
- Create hidden folder for temporary files *_temp* in the main location.
- Create a *core_validate.sas* program based on the *core_validate_data.sas*

This approach enables seamless integration of CORE validation into SAS-based clinical programming workflows while maintaining alignment with governed CDISC standards.

SUBMIFY CORE Validation Macro

The *%core_validate* macro is a central component of the SUBMIFY solution and provides integrated, standards-based validation of clinical datasets during both development and quality control (QC). Its primary purpose is to enable early and continuous conformance checking against CDISC and FDA requirements, thereby reducing downstream validation risk and improving overall data quality.
The macro supports “on-the-fly” validation of SDTM domains as they are created or updated. By identifying conformance issues early in the development lifecycle, *%core_validate* streamlines the final delivery phase. Early issue detection significantly reduces the likelihood of unexpected findings during final conformance checks.

By leveraging the CORE engine, the macro ensures alignment with current, governed standards while remaining transparent, extensible, and consistent across studies.



Usage Overview

Use of the *%core_validate* macro follows a standardized three-step process:

1. **Setup**
   A study-level setup file is configured to include the CORE configuration program, define study-specific macro variables, and make the validation macro available within the SAS session.
2. **Execution**
   The *%core_validate* macro is executed at the end of SDTM domain programs, with parameters specifying the domain, CDISC standard, and standard version.
3. **Review**
   Validation reports are reviewed, issues are addressed, and programs are rerun as needed until conformance is achieved.

We have made the use of the macro mandatory for development activities within SUBMIFY. For QC activities, use of the macro is optional but recommended as an additional conformance-checking mechanism.

Study Setup and Configuration

To enable CORE validation, the study setup file includes the CORE configuration program and defines study-specific macro variables that establish access to CORE metadata libraries and execution settings. Key macro variables include:

- core_outpath: Output location for CORE validation reports
- sdtm_version: SDTMIG version used for validation
- ct_version: Controlled terminology package version, when applicable

The values specified for sdtm_version and ct_version must correspond exactly to entries available in the CORE metadata tables. This ensures that validation is performed using a valid and supported rule set.

Macro Parameters and Validation Scope

The *%core_validate* macro supports required parameters for domain name, CDISC standard, and standard version, along with optional parameters for report timestamping and controlled terminology specification. Each invocation validates a single domain. If a corresponding supplemental qualifier dataset (SUPPxx) exists, it is automatically included in validation.

The current implementation supports SDTM validation only. Some CORE rules require related domains (e.g., DM or EX) for cross-domain consistency checks; therefore, reports may reference findings from domains not yet available. Such findings may be disregarded when outside the scope of the current validation.

Per-domain validation compared with traditional study-level validation approach

Example: Vitals (VS) Per-Domain Validation Versus Study-Level Validation

Under a traditional study-level validation approach, the Vitals (VS) domain is typically validated only after all SDTM domains have been finalized. During end-of-study validation, findings may include missing or incorrectly populated required variables (e.g., VSSTRESC or VSTESTCD), invalid controlled terminology for VSTESTCD, inconsistent use of units across subjects (VSORRESU vs. VSSTRESU), or misaligned visit timing relative to DM and SV. At this stage, correcting these issues often requires revisiting the VS derivation logic, reprocessing raw data, and re-running validation across the entire study, frequently triggering additional findings in related domains.

In contrast, a per-domain validation approach incorporates CORE validation immediately after the VS domain is created. For example, execution of the *%core_validate* macro on VS identifies missing required variables, invalid controlled terminology for vital sign tests, and unit standardization issues at the point of development. Errors such as inconsistent use of mmHg for blood pressure or missing VSSTRESN values for numeric results are detected and resolved before the VS dataset is propagated downstream.

Early validation also enables timely detection of cross-domain inconsistencies. When VS is validated after DM is available, issues such as invalid USUBJID references or missing demographic alignment are flagged early. When SV or TA domains are present, visit and timing inconsistencies (e.g., VS dates outside of subject participation periods) can be identified before finalization. Because these findings are generated while the VS programming context is still active, root causes can be quickly identified and corrected with minimal rework.

By the time study-level validation is performed, the VS domain has already passed multiple rounds of targeted, domain-level conformance checks. As a result, end-of-study validation yields significantly fewer VS-related findings, reduces the need for iterative reprocessing, and shortens the overall validation timeline. This example demonstrates how per-domain validation of VS shifts conformance checking from a late-stage, reactive process to an early, proactive quality control activity, ultimately accelerating delivery of submission-ready SDTM datasets.

Reduction in Time to Submission-Ready SDTM

The SUBMIFY CORE validation framework reduces time to submission-ready SDTM by shifting conformance checking from a late-stage, reactive activity to an early, continuous, and standardized process embedded directly within SDTM development. Integrating CORE validation at the domain level allows issues to be identified and resolved as datasets are created, rather than accumulating during final compliance checks.

Traditional SDTM workflows rely heavily on end-of-cycle validation using tools such as Pinnacle 21, often resulting in a high volume of findings late in development. Addressing these findings typically requires iterative rework across multiple domains, repeated program execution, and multiple validation cycles. In contrast, the *%core_validate* macro enables immediate, domain-specific validation, preventing the propagation of structural, controlled terminology, and rule-based errors into downstream datasets. Early validation also improves issue localization. Because findings are generated at the point of domain creation, they can be directly traced to specific programs, derivation logic, or source mappings. This significantly reduces root-cause analysis time, particularly for complex cross-domain rules, and accelerates correction cycles.

The framework's configuration-driven design further enhances efficiency by enforcing consistent use of SDTMIG versions and controlled terminology across all domains. Version misalignment is a common source of late-stage validation failures; locking these parameters at the study level minimizes the risk of rework during submission preparation.

Because CORE rules are derived from official CDISC conformance rules and FDA business rules, validation performed during development closely aligns with regulatory submission expectations. As a result, datasets that consistently pass CORE validation throughout development are more likely to pass final conformance checks with minimal additional findings. Mandatory use of *%core_validate* during development establishes a consistent quality baseline, reduces variability across programmers, and enables a more predictable and efficient path to submission-ready SDTM datasets.

Outcomes from per-domain validation were also compared with traditional study-level validation approaches. This comparison demonstrates that early, domain-level conformance checks substantially reduce both the number and severity of downstream errors, shorten overall development timelines, and promote a compliance-first mindset throughout the programming lifecycle. By embedding validation directly into routine SDTM development, teams transition from reactive issue resolution to proactive quality assurance.

Finally, the advantages of CDISC CORE over commercial validation tools, including transparency of rule logic, flexibility to incorporate custom rules, and cost-effectiveness, further support this approach. Collectively, these strengths position the proposed CORE-based framework as a scalable and sustainable solution for clinical programming teams seeking to improve efficiency while ensuring regulatory readiness.

CONCLUSION

This case study demonstrates that integrating the CDISC Open Rules Engine (CORE) directly into a SAS®-based SDTM development workflow enables a practical shift from reactive, end-of-study validation to proactive, continuous conformance checking. By implementing per-domain validation through the *%core_validate* macro, conformance issues are identified and resolved at the point of dataset creation, improving issue localization, reducing rework, and accelerating the path to submission-ready SDTM. The Vitals (VS) example illustrates how early validation materially reduces downstream findings and shortens overall validation timelines when compared with traditional study-level approaches. Beyond efficiency gains, the use of CORE promotes a compliance-first culture grounded in governed CDISC rules, transparency, and reproducibility. As an open-source, flexible, and cost-effective alternative to commercial tools, CORE provides a scalable solution for clinical programming teams seeking to enhance data quality, standardize validation practices, and improve regulatory readiness across studies.

REFERENCES

1. CDISC CORE: CDISC Open Rules Engine
   https://www.cdisc.org/core

2. CDISC Open-Source Alliance (COSA)
   https://www.cdisc.org/cosa

3. CDISC CORE Engine GitHub Repository
   https://github.com/cdisc-org/cdisc-rules-engine

4. PharmaSUG 2025, SD-044
   https://pharmasug.org/proceedings/2025/SD/PharmaSUG-2025-SD-044.pdf

5. PharmaSUG 2025, SI-380
   https://pharmasug.org/proceedings/2025/SI/PharmaSUG-2025-SI-380.pdf

ACKNOWLEDGMENTS

I would like to extend my gratitude to the entire Stats and Programming team and the Services and Solutions team at Clymb Clinical for their dedication and contributions. Their efforts have been instrumental in building an innovative solution that advances the Clymb vision of enhancing efficiency and quality in the SDTM generation, review.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Shivani Gupta
Director of Statistical Programming,
Clymb Clinical
sgupta@clymbclinical.com