Paper DS03

CDISC Open Rules: a deep dive into custom rule creation

Roman Radelicki, SGS, Belgium

ABSTRACT

The CDISC Open Rules project encourages the use of its open-source software to test study data for conformance to various standards by using the same set of unambiguous rules governed by CDISC. Next to these rules, establishing an industry-standard method for rule creation offers a promising prospect. Emphasizing the versatile application of Conformance Rules not just as submission standards, but as essential tools throughout the clinical data lifecycle, facilitating communication and collaboration among stakeholders. This paper will explore creating custom rules using the Conformance Rule Authoring tool. Our journey/challenges of implementing the authoring tool in our cloud environment and running custom rules with the CDISC Open Rules Engine (CORE) , will be shared. We'll further explore several use cases, including data listings, validation of non-SDTM clinical data with a custom define.xml, e.g. external vendor data. We will share some considerations for the future of CORE and talk about a possible AI integration.

INTRODUCTION

In the clinical research industry, the goal is to create clinical datasets that are submission-ready. Best practice involves running conformance rules to check data quality, ideally on an ongoing basis throughout the study's duration. However, a significant challenge arises from the presence of multiple organizations, each with their own interpretations of rule descriptions. These varying interpretations can potentially lead to discrepancies in the data at the time of submission. Discrepancies occur when different organizations understand rule descriptions in divergent ways. The consequence of these differing interpretations is a variety in data quality across the industry, which can undermine the uniformity and reliability of clinical data. To address this issue, it is essential for all stakeholders to align and speak the same language.

This alignment brings us to the CDISC Open Rules project — a CDISC-governed set of executable conformance rules. CDISC Open Rules serve as a single source of truth, integrating CDISC standards and FDA business rules (other regulators to follow) through a collaborative effort. It represents a community-driven initiative, supported by the CDISC Open Rules community - the Conformance Rules Development Team. The Conformance Rules Development Team ensures that the rules are correct, complete, and remain current as standards evolve. Importantly, participation in this team is not limited to programmers, developers, or data scientists but also include individuals without coding experience who can also contribute meaningfully to rule development.

CDISC's conformance rules are published in the CDISC Library and are considered part of the foundational standards. In addition to providing a governed set of unambiguous and executable conformance rules, CDISC also offers a reference implementation of an open-source execution engine for these rules.

The open-source nature of CDISC's offerings is a cornerstone of its approach. The conformance rules, along with the supporting software tools, are open, transparent, and accessible to both CDISC members and non-members, with no plans for commercialization. Distributed under the permissive MIT license, these tools can be freely used, modified, and integrated, provided the license terms and copyright notices are maintained.

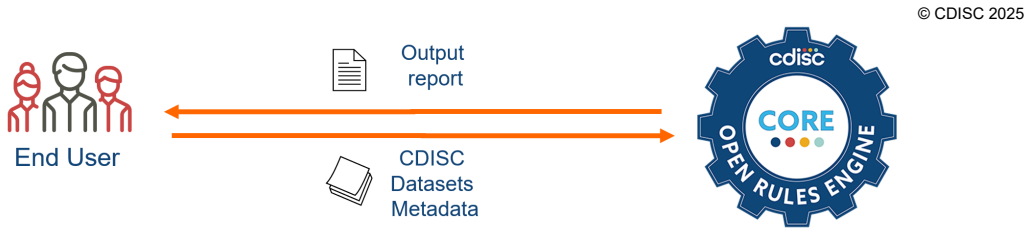
Two key tools exemplify this open-source commitment: the CDISC Open Rules Engine (CORE) and the Conformance Rules Editor. The engine is an open-source engine for validating datasets, designed to be incorporated into existing tools or used as a standalone application. The Conformance Rules Editor, on the other hand, is an intuitive tool that allows users to author conformance rules in YAML, validate them against the Conformance Rule Schema standard, test them with sample datasets, and publish them.

Through CDISC Open Rules, the industry is equipped with a robust, transparent, and collaborative framework to ensure consistency, reliability, and compliance in clinical datasets.

VALIDATE DATA USING THE CDISC OPEN RULES ENGINE

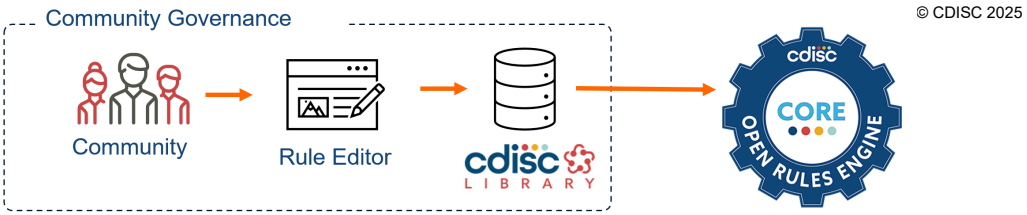
To understand how these rules and tools interact, let's explore how datasets are validated with CDISC Open Rules. The validation process begins with the engine, used by end-users such as Clinical Research Organizations (CRO), pharmaceutical companies, vendors, and regulatory agencies like the FDA. The process involves running CDISC datasets, such as SDTM along with their define.xml file, through the validation engine to produce an output report, often in formats like Excel.

Figure 1
CDISC Open Rules Engine validation of datasets



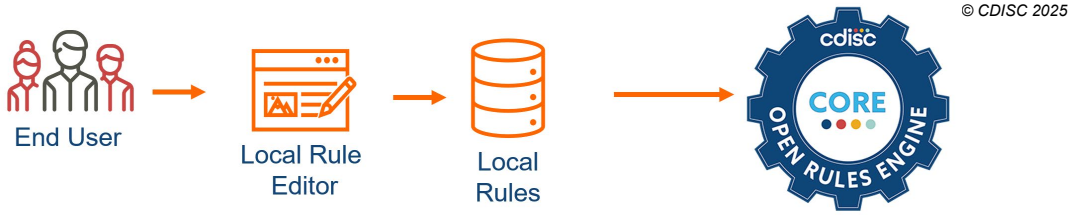
For this validation to work, the engine relies on rules fetched from the CDISC Library. These rules are created, tested, published by the CDISC community, and made available in the CDISC Library using another open-source tool—the Conformance Rules Editor. This community governance ensures that the rules remain reliable, accurate, and up to date.

Figure 2
CDISC Open Rules developing and publishing in the CDISC Library by the CDISC community



This paper goes beyond the CDISC-governed rules to explore the creation of custom rules. Specifically, it examines the possibility of establishing a local rule library and running these rules using the engine. It also addresses rules not currently included in the CDISC-governed set, as well as data cleaning rules and non-CDISC clinical data, such as vendor-specific datasets. To enable this flexibility, we also need a rule editor capable of supporting local rule creation. Naturally, the open-source Conformance Rules Editor from CDISC provides the ideal solution for this purpose.

Figure 3
Local rule development, library management and validation



THE VALUE OF LOCAL RULE CREATION

A closer look at the need for local rule creation highlights several key factors. Currently, a library of CDISC-governed rules is under active development (***Figure 4, arrow 1***), alongside ongoing collaboration with the FDA (***Figure 4, arrow 2***) to incorporate FDA business rules. These efforts represent significant progress, but the potential for broader industry collaboration remains substantial.

The establishment of an industry-standard method for rule creation holds great promise. Such a standard could open the door to new methods of interaction between various stakeholders, fostering consistency and innovation across the clinical data ecosystem.

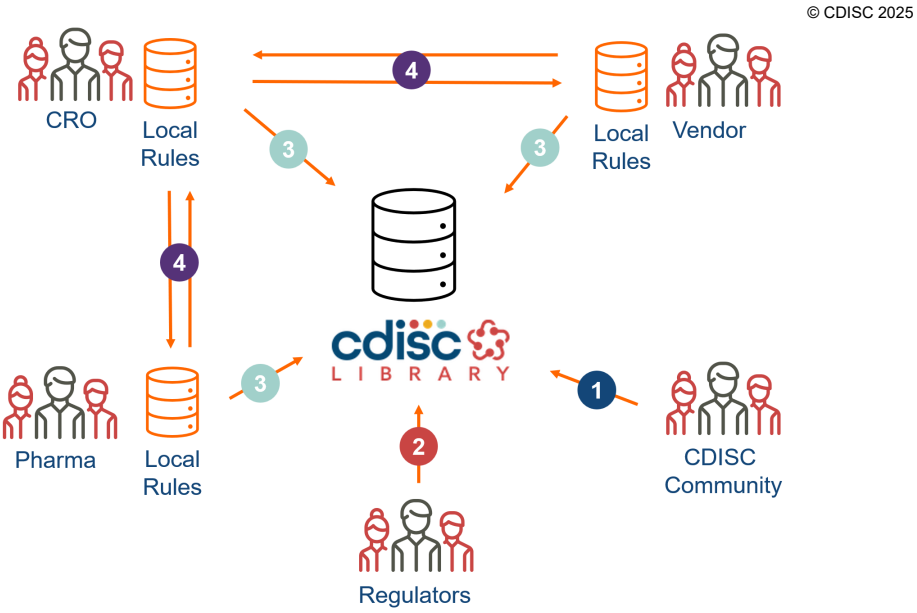
For example, a custom rule developed by one organization that proves beneficial to the broader community could be evaluated for inclusion in the CDISC-governed set (***Figure 4, arrows 3***). Additionally, there is potential for seamless transfer of custom rules between organizations (***Figure 4, arrows 4***). A pharmaceutical company might request a

CRO to create specific custom rules for improved oversight or ask the CRO to validate datasets against an existing set of company-specific rules before each transfer.

Similar interactions could also occur between other parties, such as CROs and vendors, where vendor transfers are validated against custom rules. These examples highlight just a few of the possible scenarios, but they illustrate the versatility and value of interchangeable custom rules across different stakeholders.

While this picture may not cover every possible interaction, it underscores the immense potential of establishing flexible and shareable rule frameworks within the industry.

Figure 4
Possible interactions between different stakeholders with Open Rules



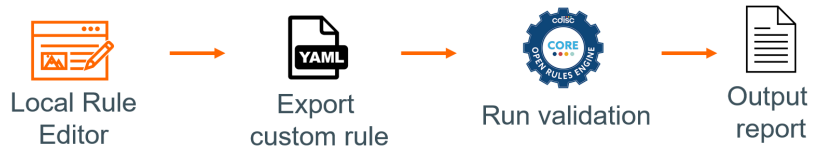
STEPS TO SUCCESSFULLY CREATE AND RUN A CUSTOM RULE

The process of creating and running custom rules involves several key steps, supported by specific tools and workflows.

PROCESS FLOW

Using a locally deployed rule editor, custom rules can be created, tested, and published. Once finalized, the rule is exported in YAML format. The engine then validates study datasets by pointing to both the datasets and the exported YAML rule file, generating a comprehensive validation report.

Figure 5
Process flow for creating local rules and running a validation



SOFTWARE REQUIREMENTS

OPEN RULES EDITOR:

The Conformance Rules Editor is a web application available for download from the CDISC GitHub repository[1]. Written in TypeScript, it features real-time syntax checking and offers a GitHub workflow for deployment in a cloud environment.

Deploying this application is not entirely straightforward and typically requires the expertise of an IT professional, such as a cloud engineer. For instance, our cloud engineer successfully deployed the editor in our Azure environment. The advantage of this setup is that it is not confined to a local installation but is instead accessible via a web browser to everyone in the organization who requires it. As a result, any team member involved in this project within clinical research can access the application and begin creating rules.

OPEN RULES ENGINE:

The engine is also available on GitHub[2], either as a Command Line Interface (CLI) or as Python source code. This flexibility allows for seamless integration into existing workflows. For example, nightly automatic conversions to SDTM datasets can include rule validation as an additional step, ensuring that newly created datasets and accompanying documentation are validated and ready for review by morning.

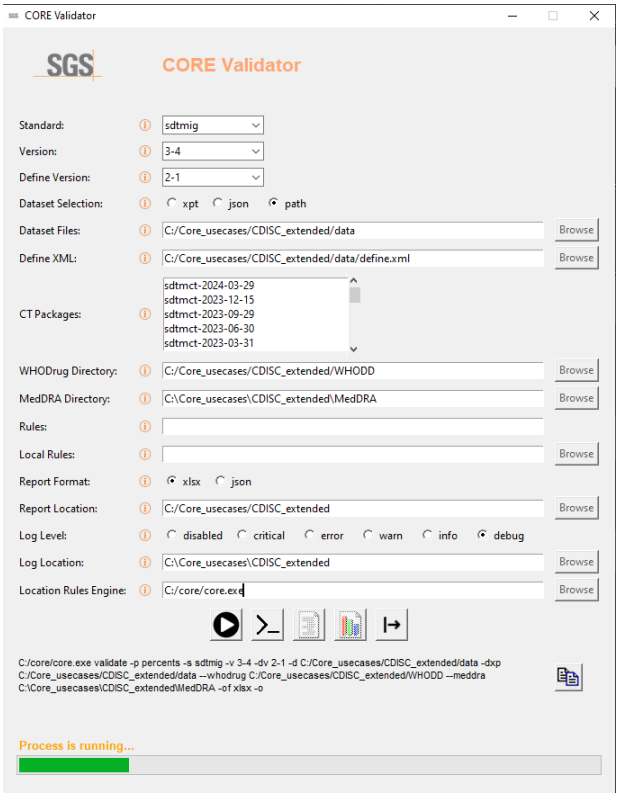
Figure 6

Clinical data conversion workflow extended with automatic CDISC CORE validation

Moreover, the engine's adaptability allows for extensions. For instance, we developed a user-friendly interface (UI) on top of the CLI, enabling less technical users to execute validation processes effortlessly. This UI translates user interactions into the appropriate engine commands, simplifying the overall workflow.

Figure 7

User-friendly interface build on top of the CDISC Open Rules Engine command line interface



USE CASES FOR CUSTOM RULES

USE CASE 1: DATA LISTINGS

A common question arises about whether CDISC Open Rules can be used to produce data listings. While this idea may sound appealing, CDISC Open Rules is fundamentally designed to flag and report violations against predefined rules, not to generate data listings. The goal of using validation rules is to minimize output, highlighting only deviations requiring attention. Listings, on the other hand, inherently generate more output.

However, the appeal of using a single tool for both checks and listings is understandable from an end-user perspective. While CDISC Open Rules is not optimized for this purpose, it remains a point of discussion for future tool enhancements.

To test this use case, I have created a custom rule designed to generate a listing of all male subjects older than 40 and female subjects older than 41.

Figure 8
Custom YAML code to produce a listing with the CDISC Open Rules

```yaml
Check:
  any:
    - all:
      - name: AGE
        operator: greater_than
        value: 40
      - name: SEX
        operator: equal_to
        value: M
    - all:
      - name: AGE
        operator: greater_than
        value: 41
      - name: SEX
        operator: equal_to
        value: F
Core:
  Id: "ROMAN-0002"
  Status: Draft
  Version: '1'
Description: in Accordance to the inclusion/exclusion criteria, Raise an error if the male subject's age is greater than 40 and the female subject's age is greater than 41.
Executability: Fully Executable
Outcome:
  Message: AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".
Rule Type: Record Data
Scope:
  Classes:
    Include:
      - SPECIAL PURPOSE
  Domains:
    Include:
      - DM
```

I provided a description and a message that will appear when the rule produces output: "AGE greater than 40 and SEX is equal to ‘M’ or AGE greater than 41 and SEX is equal to ‘F’". Additionally, I ensured that this rule applies exclusively to the DM (Demographics) domain, as indicated by the attributes Scope, Classes, Include: **SPECIAL PURPOSE** and Scope, Domains, Include: **DM.**

RULE LOGIC

The detailed rule logic can be found in the Check section, as shown at the top in **Figure 8**.
The rule is configured to produce output under the following conditions: When **SEX = M** AND **AGE > 40** OR when **SEX = F** AND **AGE > 41**. Where “any” is used for the OR operator and “all” for the AND operator.

For future trials, we might want to reuse this rule with different age thresholds. Instead of rewriting the rule entirely, it would be more efficient to parameterize these age values in a separate dataset (Trial Design) or configuration file, allowing the rule to remain flexible and reusable.

VALIDATION OUTCOME

Following the previously explained process flow for creating and validating datasets with CDISC Open Rules, I exported the rule from the editor, saved it locally, prepared the necessary test datasets, and successfully executed the validation run.

Figure 9
Output of the CDISC Open Rules validation of the custom data listing rule

CORE-ID	Message	Dataset	USUBJID	Record	Variable(s)	Value(s)
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S001	1	AGE, SEX	42.0, F
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S014	7	AGE, SEX	42.0, M
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S024	9	AGE, SEX	48.0, M
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S033	13	AGE, SEX	45.0, F
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S034	14	AGE, SEX	55.0, F
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S039	16	AGE, SEX	41.0, M
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S047	19	AGE, SEX	48.0, F
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S048	20	AGE, SEX	44.0, M
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S049	21	AGE, SEX	43.0, M
ROMAN-0002	AGE greater than 40 and SEX is equal to "M" or AGE greater than 41 and SEX is equal to "F".	DM	SGS-DRG-001-01-S052	23	AGE, SEX	51.0, M

Upon closer inspection of the validation results, the correct records were consistently present in the output file. This confirms that generating a data listing is not only feasible but also efficient. Furthermore, creating the logic for generating output proved to be highly achievable and shares many similarities with designing a standard check.

CONSIDERATIONS

The output structure is not ideal. For a listing, an Excel-style output where data can be easily filtered is often preferred. While outputting to JSON format is possible, the structure is not optimal for data listing purposes. However, with some post-transformations, it is feasible to generate an Excel-style file from the JSON output.

Figure 10
Example of how the Excel-style output for a data listing rule could look like

USUBJID	AGE	SEX	IND_COUNTRY_BEL
SGS-DRG-00A-S001	42	F	*
SGS-DRG-00A-S002	42	M	
SGS-DRG-00A-S003	48	M	
SGS-DRG-00A-S004	55	F	*
SGS-DRG-00A-S005	45	F	
SGS-DRG-00A-S006	41	M	
SGS-DRG-00A-S007	51	M	*

Potentially, we could indicate in the YAML file whether a rule is intended for a listing rather than a check. The output from a listing could then be directed to a separate report file rather than being included in the standard validation output file.

Listings often include additional context or indications, such as specifying when a subject resides in a particular region (e.g., **Figure 10** column IND_COUNTRY_BEL indicates if a subject is located in Belgium. Users can easily filter on this indication). Currently, CDISC CORE does not natively support generating such indication columns.

The key question is not whether the engine can technically support listings, but whether its scope should be expanded to explicitly support listing creation. Should this capability become a standard feature in the Open Rules, or is it something individual stakeholders (e.g., CROs, pharma companies, vendors) need to implement independently?

USE CASE 2: VALIDATION OF VENDOR DATA

Another valuable use case is the validation of vendor transfer datasets, which may not always adhere to the CDISC SDTM structure. Custom rules provide a flexible approach to address this challenge, allowing organizations to validate vendor-specific datasets effectively. This ensures data quality and compliance even when the structure deviates from CDISC standards. **Figure 11** shows a vendor test transfer in excel format which doesn't look anything like SDTM.

Figure 11
A vendor test transfer containing data in a non-CDISC SDTM structure

Trial_id	Site_id	Country	Screenid	Testname	Gender	value	Unit
Study Identifier	*Site identifier*	*Country*	*Subject Identifier*	*Lab Test Name*	*Gender*	*value of the test*	*Original Units*
Char	*Char*	*char*	*Char*	*Char*	*Char*	*Char*	*Char*
50	*50*	*50*	*50*	*50*	*50*	*50*	*50*
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S001	PROTEINE	F	4.6	g/dL
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S002	PROTEIN	F	73	U/L
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S003	PROTEINS	M	8	
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S004	CALCIUM	F	9.7	mg/dL
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S005	Magnesium	F	162	mg/dL
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S006	MAGNESIUM	F	23	U/L
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S007	GLUCOSE	M	77	mg/dL
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S008	MAGNESIUM	F	188	U/L
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S009	MAGNESIUM	F	2.1	
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S010	PHOSPHATE	M	4.4	mg/dL
SGS-DRG-001	L003	BE	SGS-DRG-001-01-S011	PROTEIN	F	7.7	g/dL
SGS-DRG-001	L004	BE	SGS-DRG-001-01-S012	ALT	M	16	U/L

Next to non-CDISC datasets, could we also use the define.xml to describe a vendor structure and, based on both the vendor data and the vendor-specific define.xml, have validation rules check the data against that define?

Creating the define.xml to represent the vendor structure is not a problem, as demonstrated in **Figure 12**. I have described all columns and even added a codelist for the expected *Testname* values.

Figure 12
Vendor specific define.xml where the structure of the expected transfer is defined including a codelist for the expected laboratory tests

LB (Labo) - [VENDOR 1.0] Location: labo.xpt

Variable	Label / Description	Type	Role	Length or Display Format	Controlled Terms or ISO Format	Origin / Source / Method / Comment
Trial_id	Study Identifier	text	Identifier	12		Protocol (Source: Sponsor)
Site_id	Site identifier	text	Identifier	2		Assigned (Source: Sponsor)
Screenid	Subject Identifier	text	Identifier	8		Assigned (Source: Sponsor)
Testname	Lab Test Name	text	Topic	6	Labo Test Code • "PROTEIN" = "Protein" • "MAGNESIUM" = "Magnesium" • "CALCIUM" = "Calcium"	Assigned (Source: Sponsor)
Gender	Gender	integer	Identifier	3		Derived (Source: Sponsor) [unresolved: MT.SEQ]
value	Result	text	Topic	6		Assigned (Source: Sponsor)
Units	Units	text	Topic	6		Assigned (Source: Sponsor)

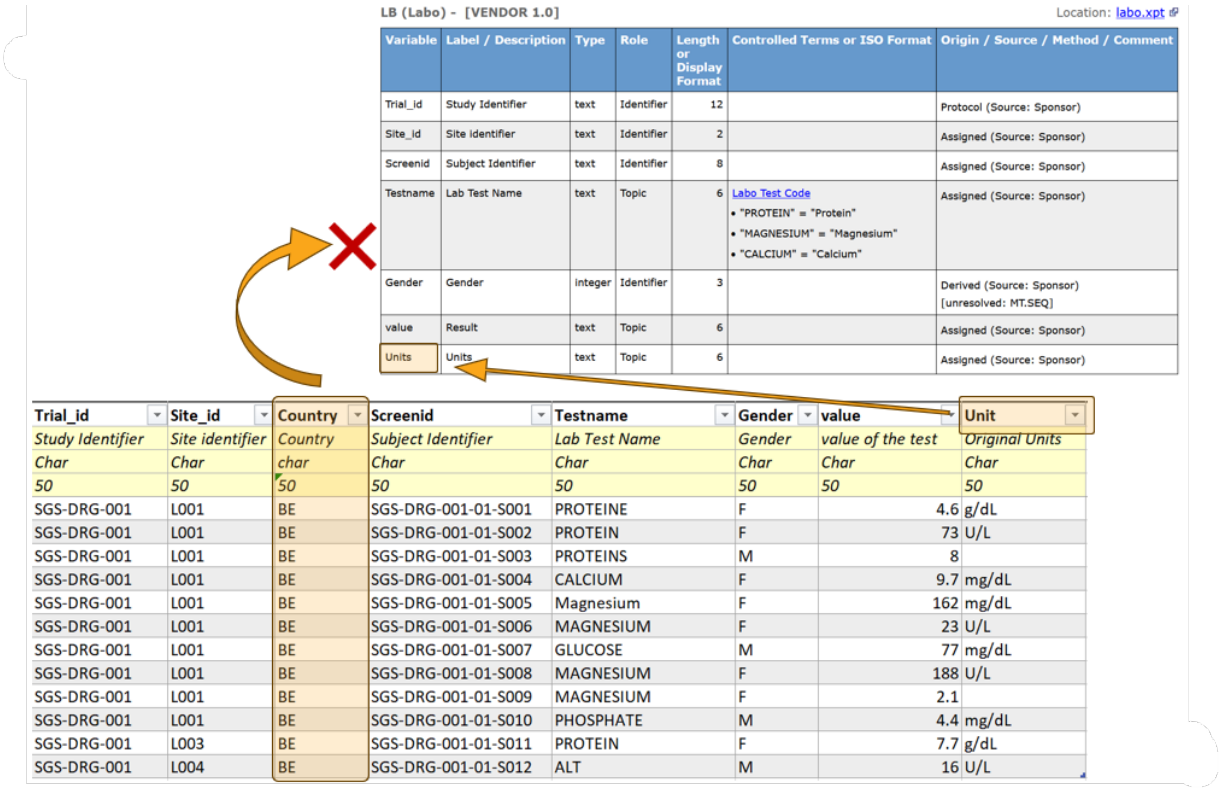
The idea would be to write a rule that validates the dataset against this define.xml. Specifically:

1. **Structure Check:** Does the received dataset contain the correct columns as specified in the define.xml?
2. **Codelist Check:** Are all *Testname* values in the dataset present in the codelist defined in the define.xml? Any values not in the codelist should appear in the validation output.

Two modifications were included to produce meaningful validation output:

- A new column named **Country** was added in the dataset.
- The **Units** column was renamed to **Unit** (removing the "s").

Figure 13
Introduction of 2 modifications to produce meaningful validation output



LB (Labo) - [VENDOR 1.0] Location: labo.xpt

Variable	Label / Description	Type	Role	Length or Display Format	Controlled Terms or ISO Format	Origin / Source / Method / Comment
Trial_id	Study Identifier	text	Identifier	12		Protocol (Source: Sponsor)
Site_id	Site identifier	text	Identifier	2		Assigned (Source: Sponsor)
Screenid	Subject Identifier	text	Identifier	8		Assigned (Source: Sponsor)
Testname	Lab Test Name	text	Topic	6	Labo Test Code • "PROTEIN" = "Protein" • "MAGNESIUM" = "Magnesium" • "CALCIUM" = "Calcium"	Assigned (Source: Sponsor)
Gender	Gender	integer	Identifier	3		Derived (Source: Sponsor) [unresolved: MT.SEQ]
value	Result	text	Topic	6		Assigned (Source: Sponsor)
Units	Units	text	Topic	6		Assigned (Source: Sponsor)

Trial_id	Site_id	Country	Screenid	Testname	Gender	value	Unit
Study Identifier	*Site identifier*	*Country*	*Subject Identifier*	*Lab Test Name*	*Gender*	*value of the test*	*Original Units*
Char	*Char*	*char*	*Char*	*Char*	*Char*	*Char*	*Char*
50	*50*	*50*	*50*	*50*	*50*	*50*	*50*
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S001	PROTEINE	F	4.6	g/dL
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S002	PROTEIN	F	73	U/L
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S003	PROTEINS	M	8	
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S004	CALCIUM	F	9.7	mg/dL
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S005	Magnesium	F	162	mg/dL
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S006	MAGNESIUM	F	23	U/L
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S007	GLUCOSE	M	77	mg/dL
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S008	MAGNESIUM	F	188	U/L
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S009	MAGNESIUM	F	2.1	
SGS-DRG-001	L001	BE	SGS-DRG-001-01-S010	PHOSPHATE	M	4.4	mg/dL
SGS-DRG-001	L003	BE	SGS-DRG-001-01-S011	PROTEIN	F	7.7	g/dL
SGS-DRG-001	L004	BE	SGS-DRG-001-01-S012	ALT	M	16	U/L

These changes allowed the rule to generate output, demonstrating the feasibility of validating vendor-specific datasets against a tailored define.xml structure.

RULE ON STRUCTURE

RULE LOGIC

The goal is to write a rule that will check if the dataset received by the vendor has the correct structure as specified in the define.xml. The key to creating this rule lies in selecting the appropriate rule type: *Variable Metadata Check against Define XML*. This choice enables a straightforward rule implementation, as illustrated in **Figure 14**, section Check. Essentially, the rule compares all dataset columns (name: variable_name) with the corresponding column definitions in the define.xml (value: define_variable_name). If any discrepancies are detected (operator: not_equal_to), they will be flagged in the output.

Figure 14
Custom YAML code to check the structure of the dataset against the define.xml

```yaml
Check:
  all:
    - name: variable_name
      operator: not_equal_to
      value: define_variable_name
Core:
  Id: "ROM-004"
  Status: Draft
  Version: '1'
Description: 'Raise an error when the variable name
in the define.xml does not
  correspond to the variable name in the dataset'
Executability: Fully Executable
Outcome:
  Message: Variable in dataset not available in the
  define.xml
Rule Type: Variable Metadata Check against Define XML
```

VALIDATION OUTCOME

In this instance, I adopted a slightly different approach to validating this rule by leveraging the editor's built-in functionality. The editor provides a feature that enables users to test a rule directly after it has been written. By clicking the *TEST* button, users can upload their define.xml file along with the corresponding vendor dataset. The editor then utilizes the engine to execute the validation and summarizes the results, as illustrated in **Figure 15**. For a more detailed analysis, clicking the *Results* button generates a comprehensive report displaying the validation output in JSON format (**Figure 16**). As anticipated, two errors were flagged: the columns *Country* and *Unit* are present in the dataset but are not specified in the define.xml.

Figure 15

Outcome of the validation on structure of the vendor transfer against the define.xml

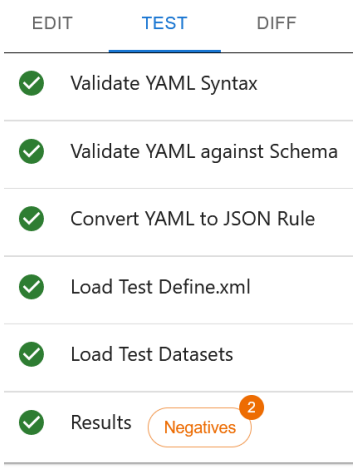


Figure 16

Validation results in JSON format showing the expected outcome

RULE ON CODELIST

RULE LOGIC

This rule aims to validate the contents of the *Testname* column in the vendor dataset, ensuring that only values specified in the codelist associated with the *Testname* variable in the *define.xml* are present (**Figure 12**). The rule logic follows a clear structure: if a variable has an associated codelist (*define_variable_has_codelist* and *define_variable_ccode* is *non-empty*), the values in the dataset must match those defined in the define.xml. Any discrepancy, where dataset values are not included in the specified codelist (*define_variable_codelist_coded_values* is_not_contained_by), will trigger an error (**Figure 17**).

Figure 17

YAML code to check the content of the Testname column in the dataset against codelist specified in the define.xml

```yaml
Check:
  all:
    - name: define_variable_ccode
      operator: non_empty
    - name: variable_value
      operator: non_empty
    - name: define_variable_has_codelist
      operator: equal_to
      value: true
    - name: variable_value
      operator: is_not_contained_by
      value: define_variable_codelist_coded_values
```

VALIDATION OUTCOME

The same validation approach using the editor was taken. The define.xml and the dataset were uploaded, and the results of the CORE validation are presented as shown in **figure 18**. **Figure 19** displays the outcome of the validation in JSON format and correctly outputs the values in the dataset that are not defined in the codelist from the define.xml.

Figure 18
Outcome of the validation on a codelist specified in the define.xml

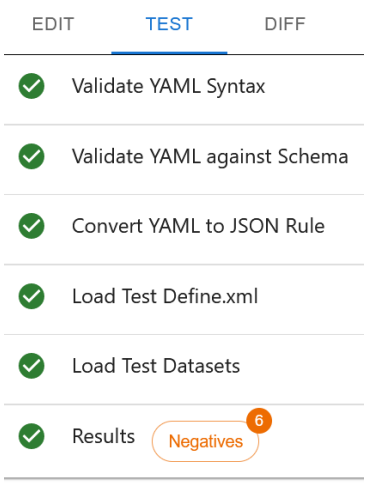
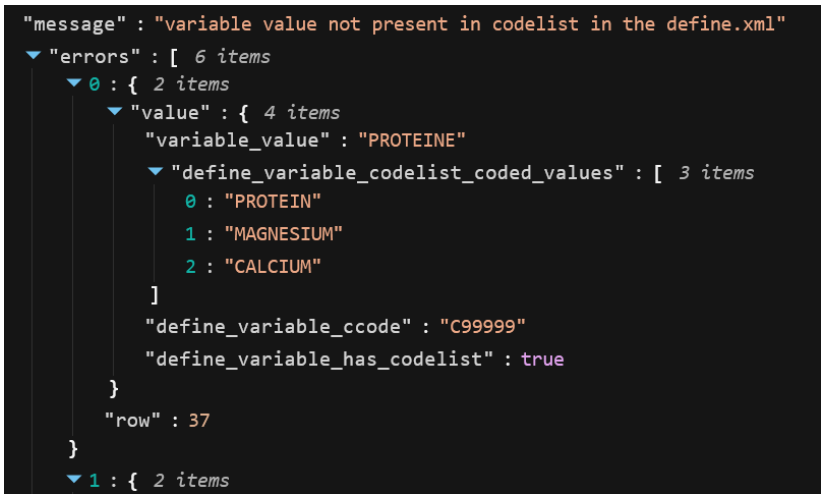


Figure 19
Part of the validation results in JSON format showing the expected outcome



This use case demonstrates that CDISC Open Rules can be effectively extended to validate external data. While this example focused on structure and codelist validation, the framework can easily be expanded to include checks on data types, value lengths, labels, value lists, and many other dimensions.

The real strength of this use case lies in leveraging a custom, non-SDTM *define.xml*. By defining validation rules against this standardized metadata file, we can create reusable rules that transcend individual trials, vendors, transfers, or standards. This eliminates the need for repeated reprogramming and ensures consistency across various datasets and stakeholders.

This approach holds significant potential. Instead of relying on static Data Transfer Agreements in Word or Excel, the *define.xml* can serve as a dynamic contract for the structure and content of data transfers. This same *define.xml* can then drive automated validation processes, as demonstrated in this use case. Beyond validation, it could further streamline downstream processes, including dataset annotation and automated mapping to SDTM.

NEXT STEPS: SCALING LOCAL RULE DEVELOPMENT FOR SUCCESS

To successfully scale local rule development across multiple teams within an organization, several key areas require focused attention:

1. GOVERNANCE AND ROLE MANAGEMENT

Effective governance is crucial when expanding rule development across teams. Clear definitions of user roles and permissions must be established to ensure smooth collaboration and avoid conflicts. Key roles may include:

- **Rule Developer:** Creates and updates rules.
- **Rule Reviewer:** Validates and approves rules before publication.
- **Rule Publisher:** Finalizes and publishes rules for use.
- **Rule Viewer:** Accesses rules without editing permissions.

Additionally, access management must be addressed: should every user have the ability to edit published rules, or should editing be restricted to specific roles? Defining these boundaries will ensure the integrity of published rules.

2. RULE CATEGORIZATION AND METADATA MANAGEMENT

The CDISC Editor was originally designed to publish rules as part of the standardized CDISC set. However, with the introduction of local rules, we now have the flexibility to create rule sets tailored for specific purposes—be it sponsor-specific, trial-specific, or vendor-specific use cases.

To manage these diverse rule sets efficiently, we need:

- A clear organizational strategy for categorizing rules.
- Enhanced metadata fields to tag rules by purpose, scope, or context.

This will enable users to easily navigate, retrieve, and apply rules across different projects.

3. VALIDATION OUTPUT MANAGEMENT
As with Pinnacle21 Community validations, each validation run generates an output report. However, not every output is equally relevant at every stage of a trial. Some findings may already have been addressed, while others may not require immediate action.

To optimize validation workflows, we need:

- The ability to assign statuses (e.g., *Open*, *Resolved*, *In Progress*) to validation findings.
- A commenting feature to document follow-up actions or rationale for decisions.
- Persistent tracking of statuses and comments across validation runs.

This approach will prevent redundant reviews of already addressed findings and improve overall efficiency.

4. EXPLORING AI INTEGRATION
As we consider the future of rule development, Artificial Intelligence presents exciting possibilities. Could a generative AI model assist in the rule editor? Potential use cases might include:

- Suggesting rule logic based on existing patterns.
- Auto-generating test data for newly created rules.

By integrating AI capabilities, we could streamline rule creation, enhance accuracy, and reduce manual effort, paving the way for smarter, faster rule development.

Such an integration of AI could take the form of a **virtual assistant** embedded directly into the rule editor, **Figure 20**. This assistant could:

- Provide real-time guidance during rule creation.
- Answer questions about rule syntax, logic, or best practices.
- Offer contextual suggestions to improve rule efficiency or clarity.
- Highlight potential errors or conflicts before validation runs.

Figure 20
A possible integration of a virtual AI assistant in the editor that can help the user with writing a rule and providing test data to validate that rule.

This idea was explored and demonstrated during the **CDISC COSA Spotlight of July 2024** (*timestamp 26:40*), which can be accessed here: https://www.cdisc.org/core/authoring-and-running-your-own-rules
In this session, a GPT named CDISC CORE Rule Generator was introduced.
A GPT[3] or a Generative Pre-trained Transformer is a custom version of ChatGPT tailored for a specific purpose—in this case, generating CDISC Open Rules.
The GPT was configured with specific instructions and linked to key reference documents, such as the CDISC Open Rules JSON schema and SDTMIG in CSV format.
This demonstration highlighted the potential of GPT models in simplifying and accelerating rule creation and validation processes. With further refinement, such AI-driven tools could become essential companions in rule development, enhancing productivity and accuracy across the board.

CONCLUSION

This paper demonstrates that developing and implementing custom local rules is not only feasible but also highly impactful. With accessible open-source software available to everyone, organizations have the tools they need to tailor validation rules to their specific needs while maintaining alignment with CDISC standards.

The use cases presented highlight the remarkable flexibility of the CDISC Open Rules framework, showcasing how rules can be adapted for diverse purposes without compromising consistency or interoperability. This interchangeability allows rules to be seamlessly shared, reused, and applied across different trials, vendors, and sponsors, reducing duplication of effort, and increasing overall efficiency.

Beyond technical benefits, this approach fosters new forms of collaboration between stakeholders. The ability to share and even donate rules back to the community creates opportunities for collective growth and continuous improvement. Furthermore, enhanced cooperation between sponsors, CROs, and vendors opens new avenues for consultancy and partnership, driving innovation and shared success across the industry.

However, the successful scaling of local rule development requires thoughtful governance and forward-looking strategies. Effective rule and user management, robust output traceability, and integration of advanced technologies—such as AI-driven virtual assistants—are all critical factors for ensuring sustainability and long-term value.

In summary, the adoption of custom local rules, supported by open-source tools, represents a significant step forward in the standardization and automation of clinical data validation. It holds the promise of greater flexibility, improved collaboration, and smarter workflows—ultimately benefiting both individual organizations and the broader CDISC community.

REFERENCES

1. https://github.com/cdisc-org/conformance-rules-editor
2. https://github.com/cdisc-org/cdisc-rules-engine
3. https://openai.com/index/introducing-gpts/

ACKNOWLEDGMENTS

Special thanks to Els, Marisa, Chris, Jeroen and Hannes for making these custom rules happen.

CONTACT INFORMATION

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

Author Name: Roman Radelicki
Company: SGS Health Science
Address:
Work Phone: +32 15 27 32 45
Email: clinicalresearch@sgs.com
Website: www.sgs.com/cro