Paper SI02

Implementing CDISC CORE in a cloud native, regulated environment supporting Rare Diseases submissions

Emmy Pahmer, OCS Consulting / argenx, 's-Hertogenbosch, Netherlands
Sandeep Juneja, argenx, Cary, USA
Stijn Rogiers, argenx, Ghent, Belgium

ABSTRACT

The focus of our presentation is to share with the CDISC community our journey implementing the CDISC Open Rules Engine (CORE) within the SAS Life Science Analytics Framework (LSAF) environment at argenx.

CORE is a free open-source tool for validating CDISC and regulatory conformance. LSAF is a closed system which provides a 21 CFR Part 11 compliant, web-based, Statistical Computing Environment (SCE) and Structured Data Repository where study data can be securely stored with versioning and audit history capabilities. Argenx is a global immunology company, committed to improving the lives of people suffering from severe autoimmune diseases.

We will highlight challenges encountered and insights gained during the implementation process. We also plan to perform an evaluation of CORE functionality, running the rules for rare disease studies. We will also share potential plans for the future in supporting the development of the CORE engine.

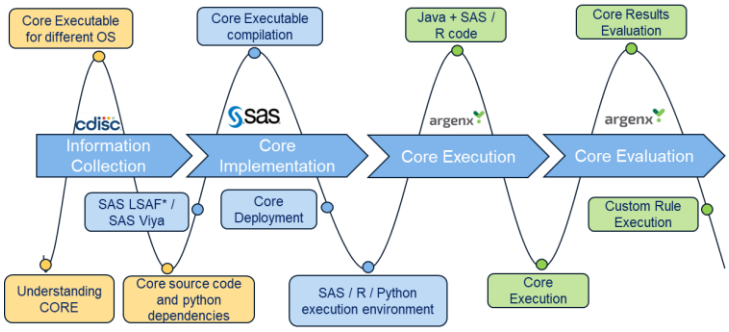
CORE INTRODUCTION

The overall goal of the CDISC Open Rules Engine (CORE) project is to deliver a governed set of unambiguous and executable conformance rules for each foundational standard.

- **Conformance Rules** - over 1900 rules covering multiple versions of implementation guides for various Foundational Standards and Regulatory Rules
- **CORE Engine** – Open-source framework, available in github and free to all CDISC community. Release the open-source engine under the CDISC Open-Source Alliance (COSA)
- **CORE Rule Editor** – Web-based application to develop CORE rules in YAML with real-time syntax checking. CORE Github - https://github.com/cdisc-org/cdisc-rules-engine

IMPLEMENTATION JOURNEY

The diagram illustrates a phased implementation journey, starting with Information Collection, moving through Core Implementation, then Core Execution, and finally Core Evaluation.



1. Information Collection

This phase included understanding the core's functionality, requirements and dependencies, for deployment across various operating systems. We used SAS applications - SAS LSAF and SAS Viya to evaluate core engine and it required deploying CDISC source code and Python dependencies in alignment with CDISC CORE requirements.

2. Core Implementation

In the Core Implementation phase, the collected source code and dependencies are compiled and integrated into a cohesive executable. Following successful compilation, the executable is deployed, and the necessary infrastructure is set up to support its operation. This phase ensures that the code is compatible with the target environments, providing a stable and reliable foundation for the core's subsequent execution.

3. **Core Execution**
To execute CDISC CORE, necessary Java and SAS code was developed to support testing and evaluation of CORE rules. Similarly, R code was also developed to evaluate CORE rules using R environment within SAS LSAF application.

4. **Core Evaluation**
During this phase CDISC CORE rules were executed and evaluated against argenx studies. Both standard rules that were part of CDISC CORE rules and also custom rules developed internally were executed and evaluated as part of this phase.

EXECUTION – SAS/JAVA

In order to execute CDISC CORE rules using SAS, custom Java code is required that needs to be wrapped into SAS macro using SAS JavaObj.

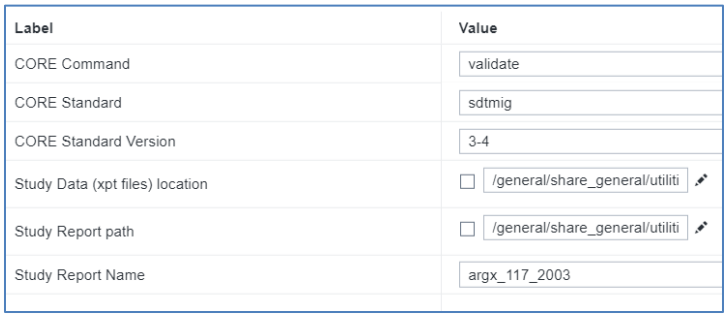
Java method: The Java method in the diagram leverages the ProcessBuilder class to execute commands with a list of parameters passed into it. It accepts arguments, such as paths, standards, and study files, which are formatted into a list and executed in a new process:

```
public void Execute(String cdisc_core_root_dir, String core_cmd, String core_standard, String core_standard_version, String resource_template, String study_xpts, String result_file) {

    List<String> params = Arrays.asList(cdisc_core_root_dir, core_cmd, "-s", core_standard, "-v", core_standard_version, "-rt",resource_template,"-d",study_xpts,"-o",result_file);
    ProcessBuilder pb = new ProcessBuilder(params);
    pb.redirectErrorStream(true);

    try {
      Process p = pb.start();
      BufferedReader reader=new BufferedReader( new InputStreamReader(p.getInputStream()));
      String s;
      while ((s = reader.readLine()) != null){
            System.out.println(s);
      }

    } catch (IOException e) {e.printStackTrace();}
}
```

SAS Macro: Below code snipped shows the java method call using the JavaObj interface within SAS. It enables SAS to execute Java code within its environment.

```
data cdisc_core_cli;
   dcl javaobj u("lsaf_cdisc_core/Run_CDISC_Core");
   u.callVoidMethod("Execute3",kstrip("&cdisc_core_root_dir/core"),kstrip("&core_cmd"), kstrip("&core_standard"),kstrip("&core_standard_version"),
                              kstrip("&resource_template"),kstrip("&study_xpts"), kstrip("&result_file"));
run;
```

LSAF (Life Science Analytics Framework) Job – It provides User Interface (UI) with ability to dynamically setup different parameter value for execution of the SAS code



It allows flexibility in executing different tasks and variations without modifying the underlying code. This setup allows for streamlined execution, with SAS handling data processing and workflow orchestration, while Java manages underlying process execution.

IMPLEMENTATION CHALLENGES

- **Different Operating Systems**: The CDISC executable was originally compiled for Windows and Linux (Ubuntu), while SAS LSAF operates on CentOS Linux. This required recompiling the CDISC source code for CentOS, ensuring all dependencies were compatible and properly installed.
- **Execution Permission**: Running the CDISC Core executable necessitated obtaining execution permissions, which involved close collaboration with the SAS team, as SAS LSAF restricts command-line execution through Base SAS.
- **Programming Language Support**: To enable Core execution in SAS, additional Java code was developed, compiled into an executable JAR, and integrated into a SAS macro. R programming language scripts were also created to facilitate Core execution within the SAS LSAF environment.
- **Shared Workspace**: The implementation needed to support a shared workspace environment, allowing multiple users concurrent access to the same executable, ensuring seamless collaboration and usage.

CORE REPORT

We ran CORE on some studies at the beginning of the year and then again in the fall. New rules and updates were implemented between the two periods. In assessing the results, we only reviewed the rules where issues were reported. The format of the output is similar to the output of other tools: an Excel file with the following tabs:

- Conformance Details – run information, which Implementation Guide, dictionaries were used.
- Dataset details – information about the datasets being tested
- Issue Summary – lists which rules generated issues and how many for each
- Issue Details – a list of each case where issues were found with USUBJID, record #, values
- Rules Report – a list of all the rules, their version, the corresponding CDISC or FDA rule, message/description of the rule and status (skipped/success)

REPORT EVALUATION – CHALLENGES

CORE rules get executed with or without a Define file.

However, if the Define is not well-formed, CORE does not execute the rules and the log provides only a general message.

Since processing is non-sequential (one check completing then the next starting), the log is difficult to read and process.

```
[INFO 2024-10-22 11:01:56,662 - console_logger.py:43] - is_suitable_for_validation. rule id=CORE-000002, domain=AE, result=False
[INFO 2024-10-22 11:01:56,662 - console_logger.py:43] - is_suitable_for_validation. rule id=CORE-000001, domain=SUPPPR, result=False
[INFO 2024-10-22 11:01:56,662 - console_logger.py:43] - Skipped domain AE.
[INFO 2024-10-22 11:01:56,662 - console_logger.py:43] - Skipped domain SUPPPR.
```

There is no information as to why rules are skipped.

CORE-000001	1	CG0176			IECAT equals "INCLUSION" and IEORRES is not equal to "N".	SKIPPED
CORE-000002	1	CG0208			SESTDTC is required.	SUCCESS

RULE LOGIC

Some of the checks did not work as expected.

Example 1.

CORE-ID	Message	Dataset	USUBJID	Variable(s)	Value(s)
CORE-000281	CMSTDTC is after CMENDTC	CM	ARGX-113-	CMENDTC, CMSTDTC	2023-08-15, 2023-08

This check is to flag when the end date is after the start date. For partial dates, it included them in the output when they were equal. This occurred in the first round of reviews and has been fixed.

Example 2.

CORE-ID	Message	Dataset	USUBJID	Variable(s)	Value(s)
CORE-000200	RSORRES cannot be null when RSTAT is null or RSDRVFL not equal to 'Y'	RS	ARGX-113-	RSDRVFL, RSORRES, RSSTAT	Y, ,

ORRES is null because the result is derived (DRVFL=Y). The check was not taking the derivation flag into consideration for this domain.

Example 3.

CORE-ID	Message	Dataset	USUBJID	Variable(s)	Value(s)
CORE-000464	QSDY is not calculated correctly even though the date portion of QSDTC is complete, the date portion of DM.RFSTDTC is a complete date, and QSDY is not empty.	qs.xpt	ARGX-113-2308-0010006005	$val_dy, QSDTC, QSDY, RFSTDTC	-1.0, 2024-07-22T09:40, 1.0, 2024-07-22T12:24

Procedure and reference date are the same. QSDY was calculated as 1 in the data. The check indicates that it should be -1.

FEEDBACK AND COOPERATION

We were able to discuss the issues we encountered with the CORE team. Rules are written on test data; it's very useful for them to receive feedback after the checks are run on production study data. Some of the issues have been corrected in the latest set of rules (example 1), and new rules are still being added. A few new issues have also been detected in the latest set of rules. (example 3).

NEXT STEPS

- **Qualification of CDISC CORE**: Since LSAF is a qualified environment, using CDISC Core in production requires passing through the argenx qualification process to meet compliance standards.
- **Automating CI/CD with the CDISC Team**: Collaborate with the CDISC team to automate continuous integration and deployment for the LSAF operating system, enhancing the efficiency and scalability of updates.
- **Involvement in CORE Engine Development**: Participate in the ongoing development of the CORE engine and its rules, ensuring alignment with current and future needs.
- **Collaboration with CRO Partners**: Engage with CRO partners to streamline workflows and align on requirements for CDISC Core, facilitating smooth integration and usage.
- **Industry Collaboration and Feedback**: Encourage a community-driven approach by collecting feedback from industry stakeholders, which will help refine and improve CORE to meet broader industry needs.

CONCLUSION

CDISC CORE is a very promising. Being able to run it in a closed, 21 CFR Part 11 compliant environment makes it even more interesting. Rules are still being developed and updated, so running the checks and providing feedback is crucial both to the CDISC CORE team and to companies submitting data to regulatory agencies.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the authors at:

Emmy Pahmer - epahmer@argenx.com / emmy.pahmer@ocs-consulting.com
Sandeep Juneja – sjuneja@argenx.com
Stijn Rogiers – srogiers@argenx.com