cdisc

2023
EUROPE
INTERCHANGE
COPENHAGEN | 26-27 APRIL

How to Extend and Run CORE

Sam Hume, DSc
CDISC
VP, Data Science

Session 6 Track B: CORE Implementation

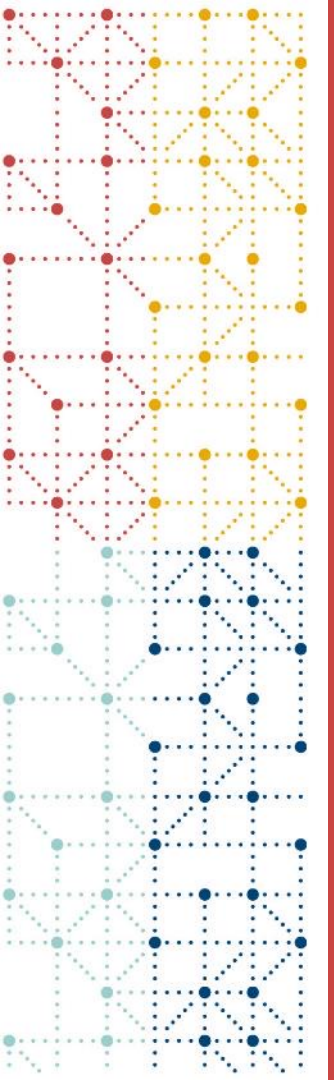
Meet the Speaker

Sam Hume

Title: VP, Data Science

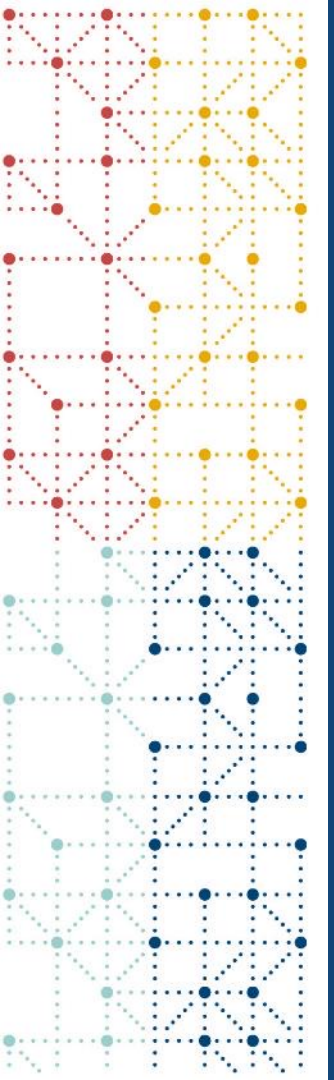
Organization: CDISC

Sam Hume leads the CDISC Data Science team, which collaborates with CDISC staff and stakeholders to develop tools and standards that support clinical and translational data science. Sam directs delivery of the CDISC Library metadata repository that houses all CDISC standards, co-leads the CDISC Data Exchange Standards team, co-leads CORE, and leads the technical CDISC RWD efforts. He has 25 years' experience in clinical research informatics and has held a number of senior technology positions in the biopharmaceutical industry. He holds a doctorate in information systems.



Agenda

1. CORE Software Overview
2. Extending the CORE Engine
3. Creating an Example Extension
4. Running the CORE Engine
5. Using the CORE Engine



Overview

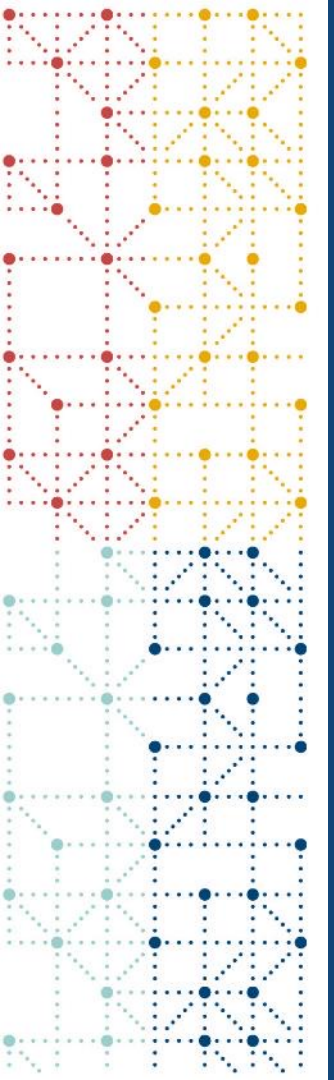
A bit of background information to get us started

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



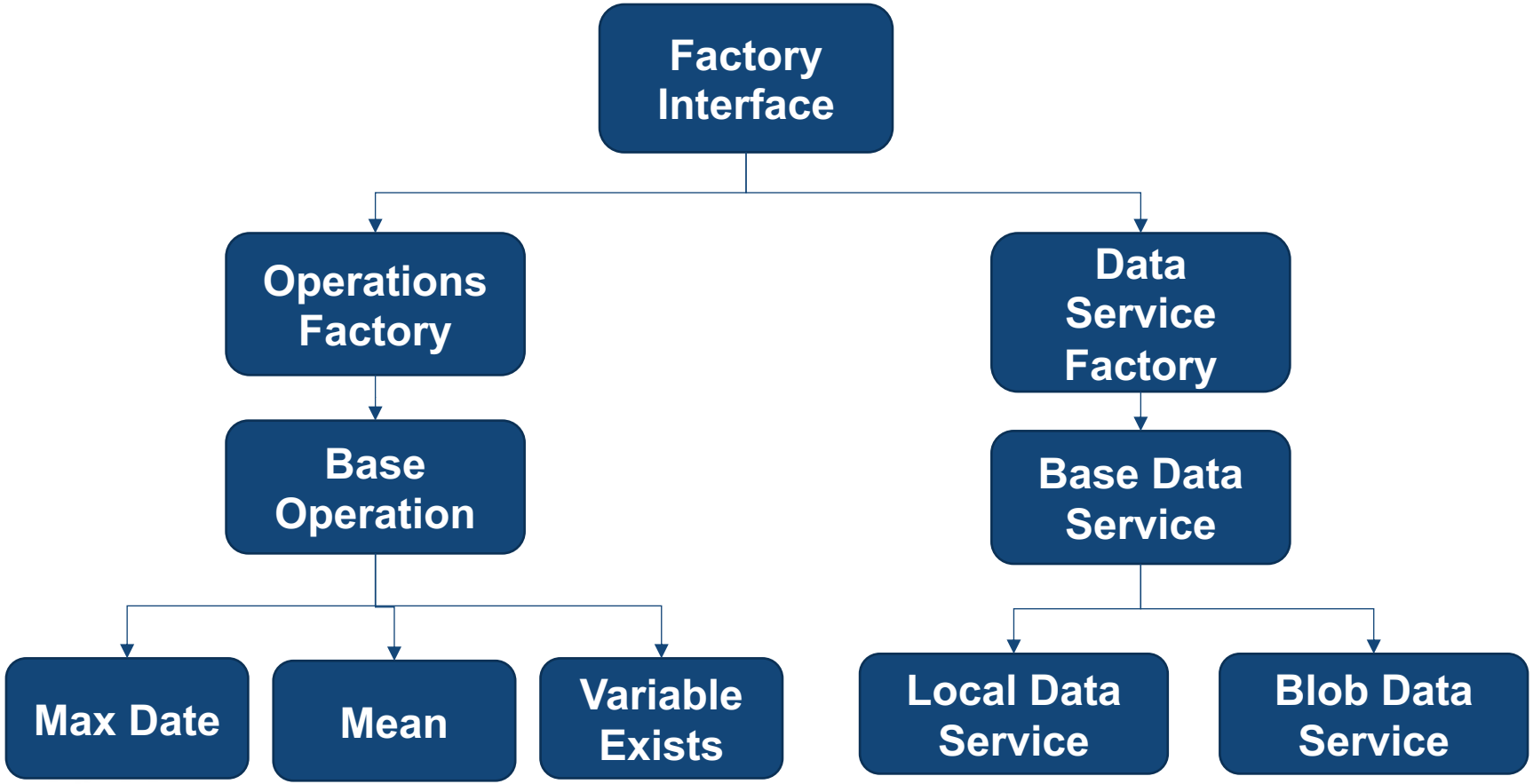
Extending the CORE Engine

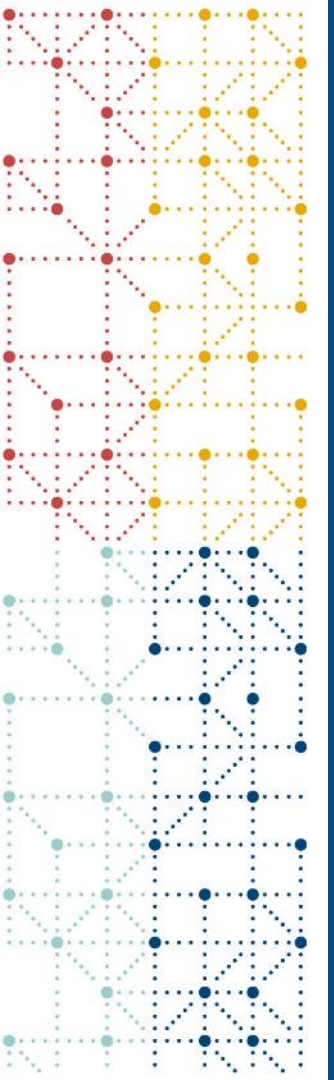
How to extend the CORE Engine

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

Creating your own operations and data services





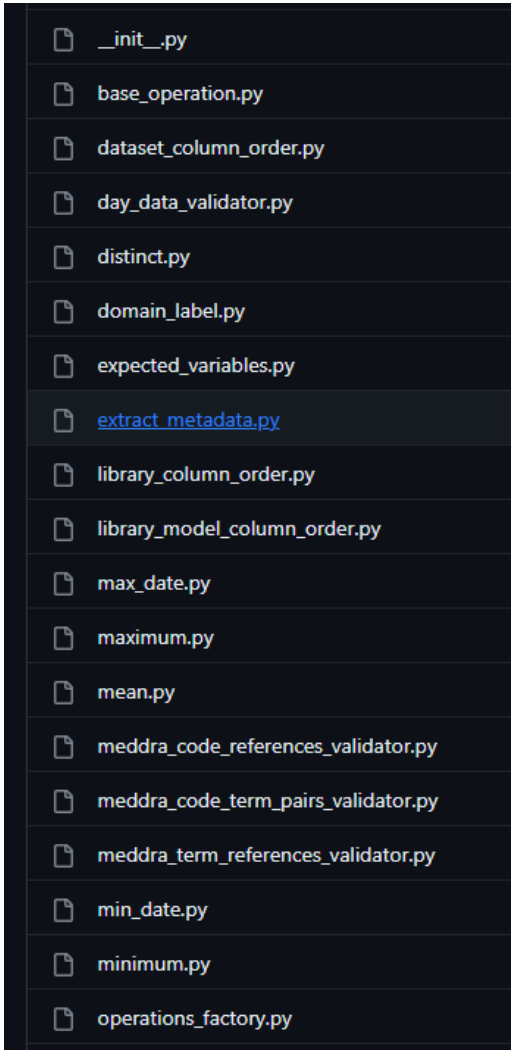
Creating an Example Extension

Creating a new operation for use in conformance rules

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

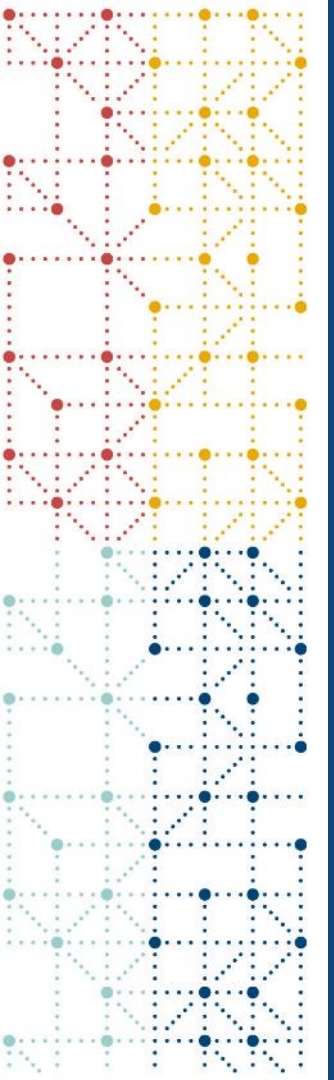
- The is_odd operation is used to create a new column that contains “true” if AGE is an odd number
- The Check examines “all” records to find cases where the $age_is_odd column equals “true”
- A report is generated identifying cases where this rule fired
- This could have been implemented as a check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

Results from executing the rule

- The test dataset contains 2 subjects with ages: 26 and 27.
- The rule fired for the subject with AGE = 27
- Running engine with this single rule generates an Excel report (bottom)
  - CORE-Report-2023-03-25T08-56-38

	A	B	C	D	E	F	G	H	
1	RuleID	Message	Executability	Dataset	USUBJID	Record	Sequence	Variable(s)	Value(s)
2	CORE-000127	Why is your age odd?	fully executable	DM	CDISC-TEST-002	2		$age_is_odd, AGE	True, 27



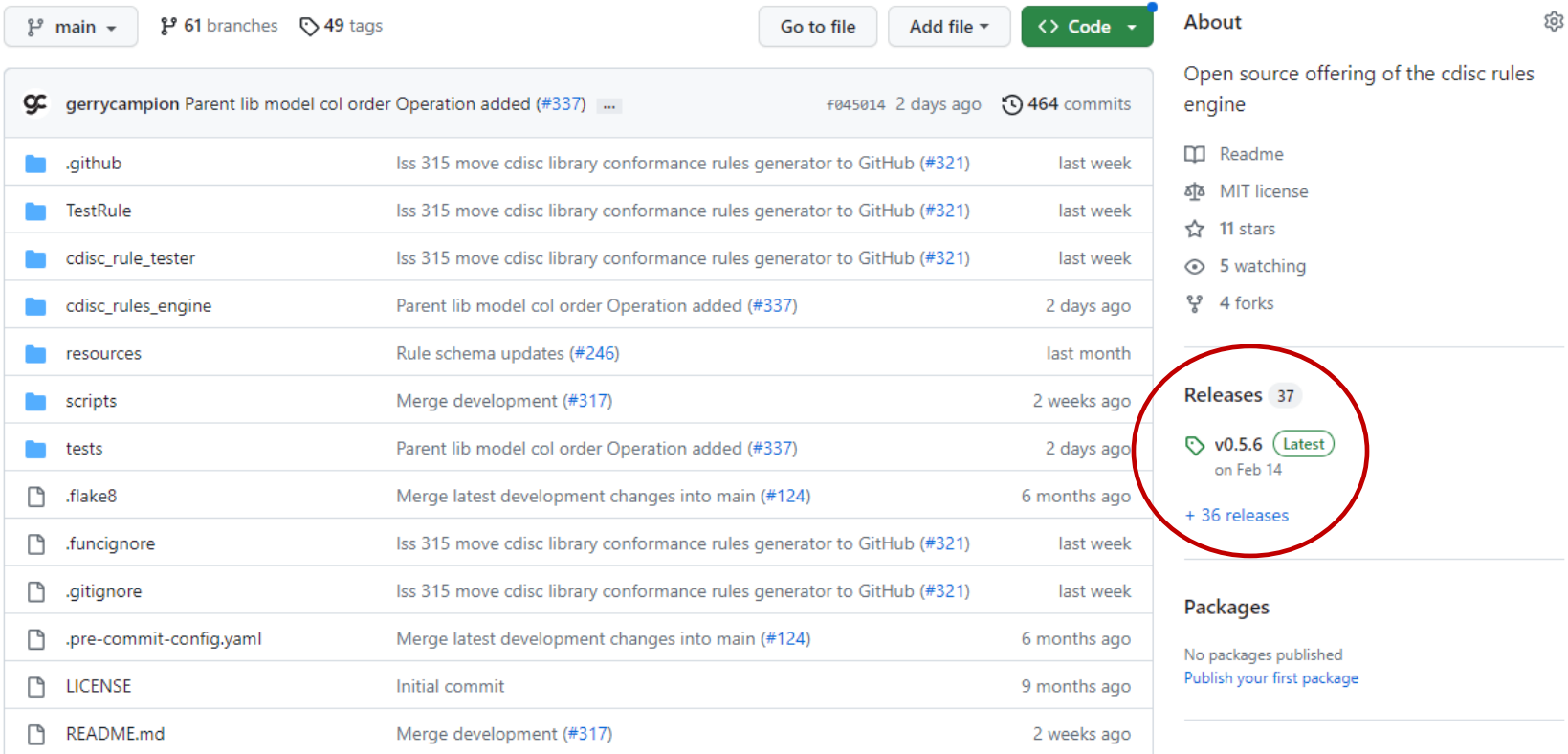
Running the CORE Engine

How to run the CORE Engine today

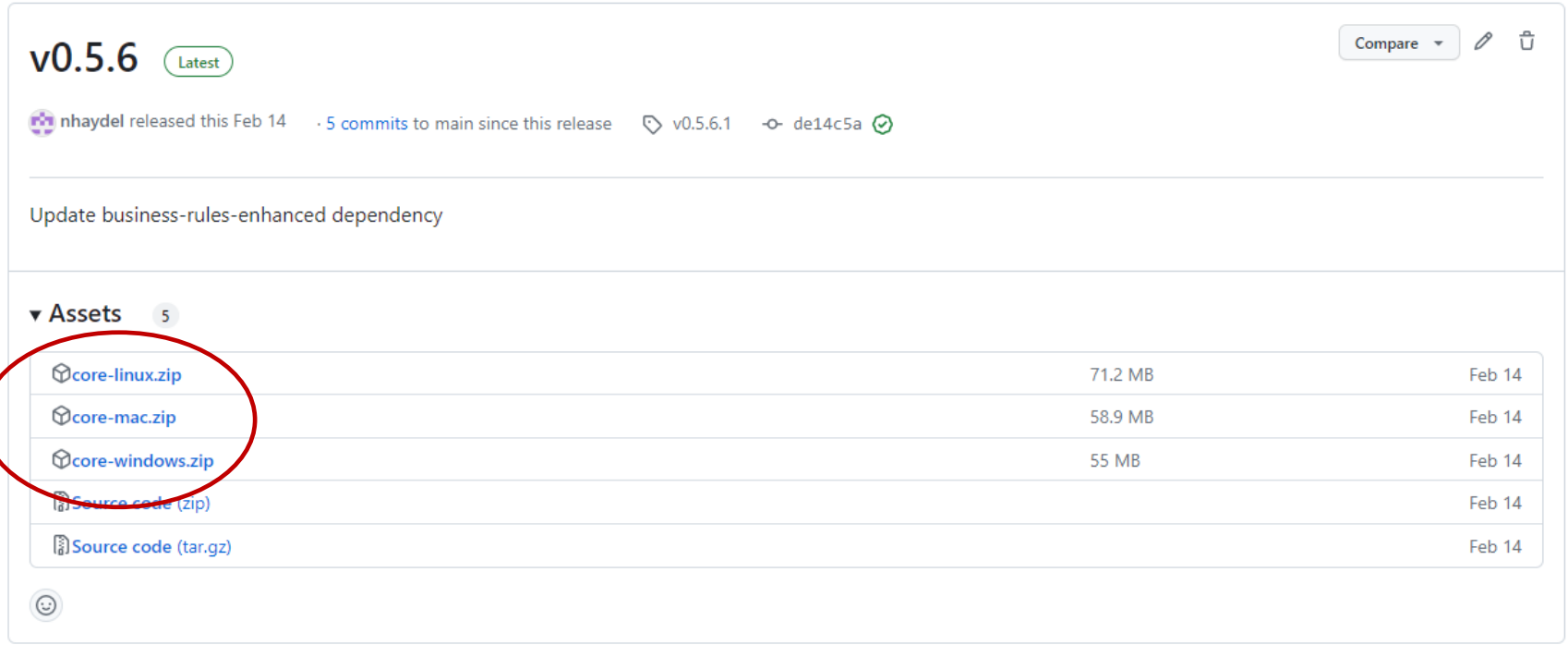
Running the CORE Engine

- CLI executable available in GitHub
  - Cached rules
  - Windows, Mac, and Linux install packages
  - Unzip and run
  - Will need datasets to validate
- Engine available on PyPI
  - Engine is a component that can be used in your own code
- Desktop versions
  - Vendor released versions of CORE
  - Includes a user-friendly UI
  - Easier for non-technical users to evaluate
- View a short CORE demonstration
  - https://www.cdisc.org/core
  - See **CORE on GitHub** tab

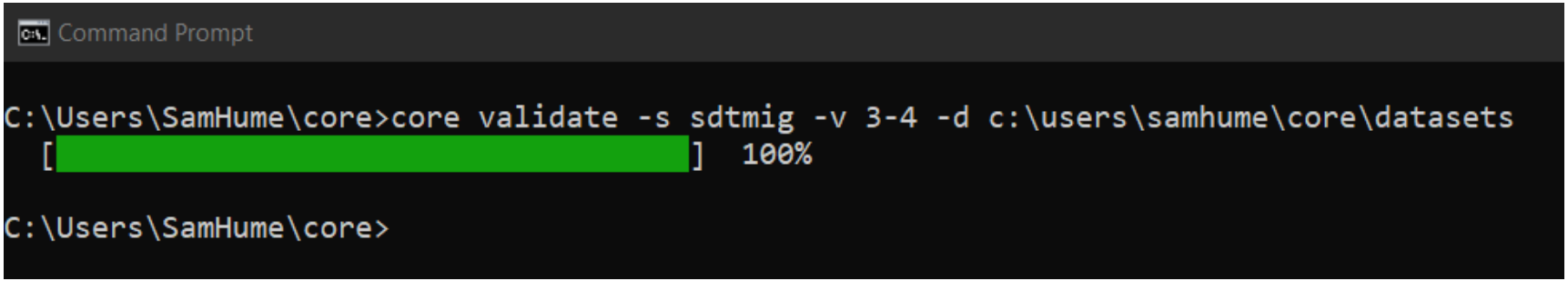
CLI Deployment – in GitHub under Releases



Download the latest CLI CORE 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

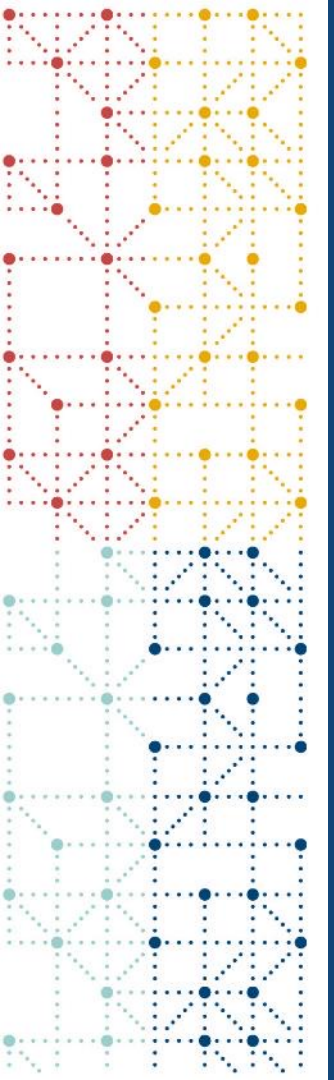
CORE report generated by the test run

	A	B
1	Conformance Details	
2	Data Location	c:\users\samhume\core\datasets/ae.xpt, c:\users\samhume\core\datasets/dm.xpt, c:\users\samhume\core\datasets/ds.xpt, c:\users\samhume\core\datasets/lb.xpt, c:\users\samhume\core\datasets/suppae.xpt
3	Report Generation	2023-03-23T13:01:08
4	Total Runtime	7.19 seconds
5		
6		
7	Bundle Details	
8	Standard	SDTMIG
9	Version	V3.4
10	CT Version	
11	Define-XML Version	
12	UNII Version	Not configured
13	Med-RT Version	Not configured
14	MedDRA Version	
15	WHODRUG Version	
16	SNOMED Version	Not configured

CORE Report

- Generated in Excel
- Placed in the CORE folder
- Datetime stamp in name

	A	B	C	D	E	
1	Dataset	RuleID	Message	Severity	Issues	Explanation
2	AE	CORE-000022	At least one of the Seriousness criteria (AESCAN, AESCONG, AESDISAB, AESDTH, AESHOSP, AESLIFE, AESOD or AESMIE) = 'Y', but AESER = 'N' or empty.		27	
3	AE	CORE-000206	AESTDTC is present in a Findings general observation class		1	
4	DM	CORE-000253	DTHFL is not 'Y' when AE.AEOUT is 'FATAL'.		9	
5	DS	CORE-000206	DSSTDTC is present in a Findings general observation class		1	
6	SV	CORE-000206	SVSTDTC is present in a Findings general observation class		1	
7	TS	CORE-000107	An appropriate subject identifier is not present. APID is required in all Associated Persons Data. In addition to STUDYID, DOMAIN, and TSSEQ being required for all domains based on one of the 3 general observation classes, one of USUBJID, APID, SPDEVID, or POOLID must also be present.		1	



Using the CORE Engine

Thoughts on deploying and using CORE

CORE Engine Deployments

CDISC Provides

- Code repository in GitHub
- CLI executable version
- PyPI library
- Stable and Development releases
- Base testing and validation package
- Respond to reported issues

Vendor Provides

- Validated production versions
- Desktop CORE with UI
- Web-based CORE package with UI
- Cloud-based deployments
- Integrated into vendor platforms
- May provide
  - Hosting
  - Support
  - Rule development
  - Complete validation package

Using CORE

- Anticipate many will deploy CORE in multiple ways
  - Using CORE within a vendor's platform
  - Setting up CORE to run in your organizations cloud environment
  - Running a desktop version of CORE
  - Running the command-line version of CORE
  - Running the CORE rules using alternative engines
  - Building tools that incorporate the CORE Engine
- CORE can be run at no cost allowing organizations to have 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

Thank You!

Sam Hume

shume@cdisc.org