

How to Automate Validation with Pinnacle 21 Command Line Interface and SAS®

Amy Garrett, Pinnacle 21, Blue Bell, USA

ABSTRACT

Pinnacle 21 software is widely used by the industry for helping prepare data in a standardized format that is required for regulatory submissions. Pinnacle 21 Command Line Interface (CLI) allows automation of validation jobs resulting in higher reliability of results and better overall performance of data preparation process. This presentation will review new capabilities and syntax of the recently updated version of Pinnacle 21 CLI and provide examples for automating validation using SAS® programming language.

INTRODUCTION

Pinnacle 21 Validator can be run either from a graphical user interface (GUI) or command line interface (CLI). Both options support the full set of validation features but are designed for different use cases. Where GUI is most commonly used for ad-hoc validation, the CLI is typically utilized for process automation and customization. Because the CLI can be called directly from SAS, it eliminates the need for the SAS programmer to exit the programming environment to run Pinnacle 21. The CLI can be kicked off periodically or when new data has been placed in a specific location or simply appended to the end of a SAS program as a QC step. Moreover, the SAS code used to call the CLI can be incorporated into a SAS macro or batch file and used over and over again for multiple studies across your organization, ensuing consistency in the validation process wherever appropriate. The CLI also offers greater power over the validation process, giving the user control over items such as validation report name and output location.

To download the newest Pinnacle 21 client, visit <https://www.pinnacle21.com/downloads> and to learn more about the CLI, go to <https://www.pinnacle21.com/documentation>

CLI PARAMETERS AND SYNTAX

The CLI has long been a feature of Pinnacle 21 products and the latest software release includes new syntax and more options. The most used parameters for all users are depicted in Table 1 and a full list can be found by running the following code from the command line:

```
java -jar -p21-client-{version}.jar -help
```

NOTABLE CHANGES

The new CLI is called from `p21-client-{version}.jar` unlike prior releases which used task-specific files: `validator-cli-{version}.jar`, `data-converter-{version}.jar`, `define-generator-{version}.jar`. Additional changes include the following:

- Support for validation of `define.xml` file has been added
- The syntax has been changed from `-parameter=value` to `--parameter=value`
- The `report:overwrite` parameter has been decommissioned and all reports will overwrite an existing report with the same name by default
- The `config` parameter is no longer required for validation and instead the user may elect to specify `standard` and `standard.version`; however, the parameter is still available for use

PhUSE EU Connect 2018

Parameter	Valid Values	Default	Description
Standard and Version			
--standard	SDTM SEND ADaM		
--standard.version	<version>		Version of standard.
Validator Source Data			
--source.adam	<File List or Path>		Path to source ADaM data files. Can contain directories or individual files separated by semicolon.
--source.sdtm	<File List or Path>		Path to source SDTM data files. Can contain directories or individual files separated by semicolon.
--source.send	<File List or Path>		Path to source SEND data files. Can contain directories or individual files separated by semicolon.
--source.define	<File>		Path to define file.
Validator - Terminology			
--cdisc.ct.adam.version	<version>	<latest>	Version of ADaM controlled terminology.
--cdisc.ct.sdtm.version	<version>	<latest>	Version of SDTM CDISC controlled terminology.
--cdisc.ct.send.version	<version>	<latest>	Version of SEND CDISC controlled terminology.
--meddra.version	<version>	<latest>	Version of MedDRA dictionary.
--ndf-rt.version	<version>	<latest>	Version of NDF-RT dictionary.
--snomed.version	<version>	<latest>	Version of SNOMED dictionary.
--unii.version	<version>	<latest>	Version of UNII dictionary.
Validator - Report			
--report	<File or Path>		File or path to Excel specification. If Path is provided, CLI will use default file name [pinnacle21-report-yyyy-MM-ddTHH-mm.xlsx] --report parameter is required for data validation.
--report.cutoff	<#>	1000	Number of times a record detail is printed per issue. For no limit specify 0.
Define.xml Designer - Convert Excel to Define.xml			
--source	<File or Path>		
--output	<File or Path>		Either --report or --output parameter is required.

Table 1. List of Commonly Used CLI Parameters

CALLING THE CLI FROM SAS

The latest CLI is a robust tool and can easily be integrated into SAS programs. In the following examples, the core features of the CLI are depicted by calling the CLI using the X statement in SAS. Let's start by assigning some key pieces of information to macro variables so we can reference them later:

PhUSE EU Connect 2018

```
/* Pinnacle 21 Parameters */
/* Note: the configuration file folder should be in the same location as p21-client-
{version}.jar */

%let jarpath = C:\Users\p21-client ;
%let jarfile = p21-client-1.0.7.jar ;
%let configpath = C:\Users\p21-client\config ;

/* CDISC Data Parameters */
%let sdtmpath = C:\Users\study-101\sdtm\xpt ;
%let adampath = C:\Users\study-101\adam\xpt ;
%let sdmtdefine = C:\Users\study-101\sdtm\Define ;
%let adamdefine = C:\Users\study-101\adam\Define ;

/* Validation Report Parameters */
%let reportpath = C:\p21-client\reports;
```

Example 1. SDTM Validation

```
x
java -jar "&jarpath.\&jarfile." ^
--standard=sdtm ^
--standard.version=3.2 ^
--source.sdtm="&sdtmpath." ^
--source.define="&sdmtdefine." ^
--cdisc.ct.sdtm.version=2017-12-22 ^
--unii.version=2017-11-15 ^
--ndf-rt.version=2018-02-05 ^
--meddra.version=19.0 ^
--report="&reportpath.\s101-sdtm.xlsx "
;
```

Example 2. ADaM Validation

```
x
java -jar "&jarpath.\&jarfile." ^
--standard=adam ^
--standard.version=1.0 ^
--source.sdtm="&sdtmpath." ^
--source.adam="&adampath." ^
--cdisc.ct.sdtm.version=2017-12-22 ^
--cdisc.ct.adam.version=2017-09-29 ^
--meddra.version=19.0 ^
--report="&reportpath.\s101-adam.xlsx"
;
```

Example 3. ADaM Validation (ADSL and ADAE only)

```
x
java -jar "&jarpath.\&jarfile." ^
--standard=adam ^
--standard.version=1.0 ^
--source.adam="&adampath.\ADSL.xpt;&adampath.\ADAE.xpt" ^
--cdisc.ct.sdtm.version=2017-12-22 ^
--cdisc.ct.adam.version=2017-09-29 ^
--report="&reportpath.\s101-adam-adae.xlsx"
;
```

Example 4. Define Template Generator

```
x
java -jar "&jarpath.\&jarfile." ^
--standard=sdtm ^
--standard.version=3.2 ^
```

PhUSE EU Connect 2018

```
--source.sdtm="%sdtmpath." ^
--meddra.version=19.0
--cdisc.ct.sdtm.version=2017-12-22 ^
--output.format=define-excel ^
--output="%reportpath.\sdtm_define_template.xlsx"
;
```

Example 5. Define.XML Generator

```
x
java -jar "&jarpath.\&jarfile." ^
--standard=sdtm ^
--standard.version=3.2 ^
--source="%reportpath.\sdtm_define_template.xlsx" ^
--output="%reportpath.\sdtm_define.xml" ^
;
```

Example 6. Define.XML Validation

```
x
java -jar "&jarpath.\&jarfile." ^
--config="%configpath\Define.xml" ^
--standard=sdtm ^
--source.define="%sdtmdefine.\define.xml" ^
--cdisc.ct.sdtm.version=2017-12-22 ^
--report="%reportpath.\define_validation_report.xlsx"
;
```

VALIDATION RETURN CODES

When the `report` parameter is provided, CLI will perform high level analysis of the validation report and returns a code for the highest severity found on the Issue Summary sheet. For example, if there are many issues reported and one of them is a Reject, the CLI return code will be 1. A list of possible return codes are shown in Table 2. The return code is stored in `%errorlevel%` and `$result` for Windows and Mac systems, respectively. Example 7 shows a batch program that calls the CLI to perform SDTM validation and outputs the return code to a file. Saving the return code to a file may be useful for signaling the start of additional workflows. For example, if the SDTM CLI return code is greater than two, ADaM validation can automatically start

Return Code	Severity Value
1	Reject
2	Error
3	Warning
4	Notice
5	N/A

Table 2. List of CLI Return Codes

Example 7. Calling CLI from Windows batch file and output CLI return code to file

```
java -jar %jar% --standard=sdtm --standard.version=3.2 --source.sdtm=%sdtmdata% --
cdisc.ct.sdtm.version=2017-12-22 --meddra.version=19.0 --report=%valreport%
@echo errorlevel=%errorlevel% >> "C:\s101-sdtm-error.txt"
```

ENTERPRISE USE

Enterprise users can call the CLI to perform tasks over an Enterprise server by providing additional parameters. Enterprise-specific parameters are listed in Table 3. The additional parameters can be added to existing code or specified in the configuration file, `pinnacle21.conf`. The `pinnacle21.conf` file should be saved in the same file location as `p21-client-{version}.jar` and look similar to the sample configuration file presented in Figure 1. It is recommended to place static parameters that deal with connectivity or environment in the configuration file to keep command line clean and relevant only to the project. After the configuration file is set-up for use, the CLI will automatically pull the specified parameters from the file and run in "Enterprise mode". All validations are stored on the Enterprise server and access to validation results can be controlled from the CLI using the `group` parameter such that all users assigned to the specified group will be able to see the validations completed using the CLI.

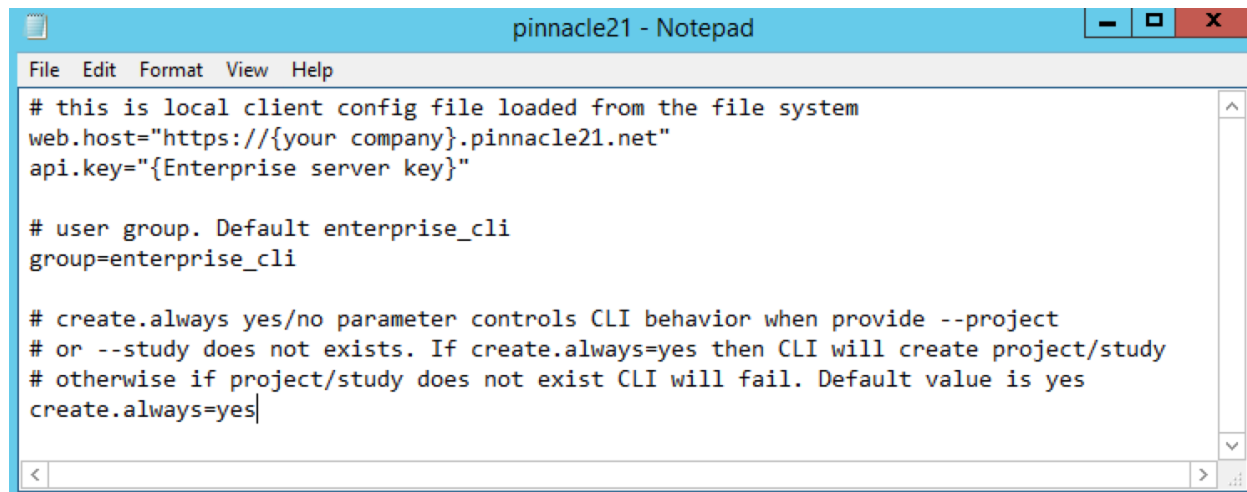
Example 8 is an SDTM validation modified to run on an Enterprise server and pulls Enterprise parameters from

PhUSE EU Connect 2018

pinnacle21.conf. Example 9 is an ADaM validation that specifies the Enterprise parameters in the command line instead of the configuration file. Transitioning from Community CLI to Enterprise CLI is easy and the resulting enterprise validation report includes more information as depicted in Figure 2.

Enterprise Parameter	Valid Values	Req	Default	Conf. file	Description
Enterprise Connection Parameters					
--api.key	API Key	x		x	API Key from the enterprise server API Keys admin page
--web.host	URL	x		x	Enterprise server URL
--group			enterprise_cli	x	User group. Should be created on the server prior to using CLI
Enterprise Project Setup Parameters					
--project	Project Name		enterprise_cli	x	Name of the project. CLI creates new project if does not exists.
--study	Study Name		enterprise_cli	x	Name of the study. CLI creates new study if does not exists.
--filter	ALL PMDA FDA		ALL	x	Config filter to apply.
--datapackage	Data package name				Name of the datapackage. CLI creates new data package if does not exists.
--create.always	Yes No		Yes	x	Create project and/or study based on command line --project/--study parameters if does not exist. Default is Yes. To prevent auto create set this parameter to No.

Table 3. List of Enterprise-Specific CLI Parameters



```

# this is local client config file loaded from the file system
web.host="https://{your company}.pinnacle21.net"
api.key="{Enterprise server key}"

# user group. Default enterprise_cli
group=enterprise_cli

# create.always yes/no parameter controls CLI behavior when provide --project
# or --study does not exists. If create.always=yes then CLI will create project/study
# otherwise if project/study does not exist CLI will fail. Default value is yes
create.always=yes

```

Figure 1. Sample configuration file, pinnacle21.conf

PhUSE EU Connect 2018

Example 8. SDTM Validation with Enterprise parameters from pinnacle21.conf

```
x
java -jar "&jarpath.\&jarfile." ^
--standard=sdm ^
--standard.version=3.2 ^
--source.sdm="aspera://&sdtmpath." ^
--source.define="&sdmdefine." ^
--meddra.version=19.0 ^
--report="&reportpath.\s101-sdm.xlsx "
;
```

Example 9. ADaM Validation with Enterprise parameters from the command line

```
x
java -jar "&jarpath.\&jarfile." ^
--study="s101" ^
--group="s101_team" ^
--web.host="&url." ^
--api.key="&apikey." ^
--standard=adam ^
--standard.version=1.0 ^
--source.sdm="aspera://&sdtmpath." ^
--source.adam="aspera://&adampath." ^
--meddra.version=19.0 ^
--report="&reportpath.\s101-adam.xlsx"
;
```

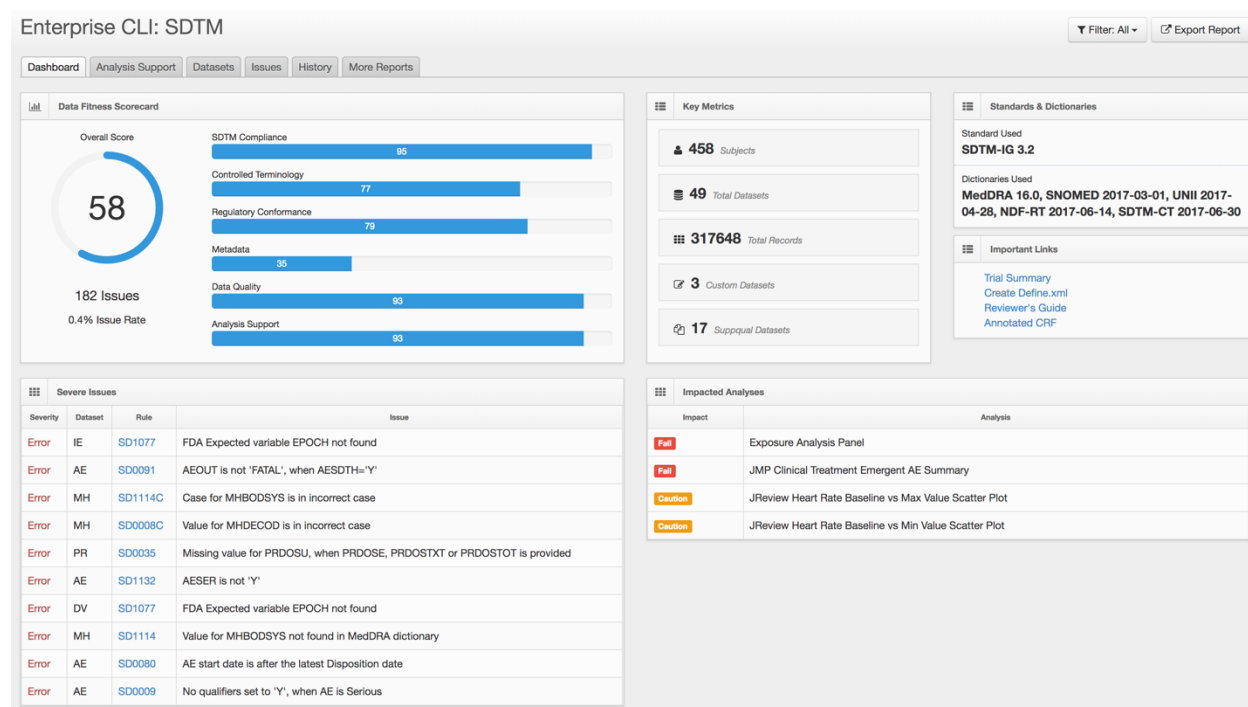


Figure 2. Enterprise Validation Generated with CLI

ERROR HANDLING AND DEBUGGING

Running Pinnacle 21 with the CLI is simple but if something doesn't work as expected, a list of error codes are provided in Table 3 to help you debug and identify any issues. After using the CLI, a log file, `p21-client-{version}.log` is created. Check this log file for errors and modify the program accordingly. You can also write additional information to the log by adding the statement, `-Dcli.log.level=debug` such that the first line of the command line code is:

```
java -Dcli.log.level=debug -jar..
```

PhUSE EU Connect 2018

Error Code	Description
–CLI.1.1 [Error code: 255]	General CLI exception
–CLI.2.2 [Error code: 234]	Standard is unknown
–CLI.2.3 [Error code: 233]	Format type is unknown
–CLI.2.4 [Error code: 232]	Terminology value is invalid or not defined
–CLI.2.5 [Error code: 231]	Dictionary value is invalid or not defined
–CLI.2.6 [Error code: 230]	Enterprise data type is unknown
–CLI.2.7 [Error code: 229]	Release version is undefined
–CLI.2.8 [Error code: 228]	Parsing failed. Cannot read data source
–CLI.2.9 [Error code: 227]	Provided Value is unknown
–CLI.2.10 [Error code: 210]	Request type value is unknown
–CLI.3.1 [Error code: 225]	Request type cannot be determined
–CLI.3.2 [Error code: 224]	Failed to copy data source
–CLI.3.3 [Error code: 223]	Define file does not exists
–CLI.3.4 [Error code: 222]	Failed to copy define XML
–CLI.3.5 [Error code: 221]	Failed parsing input argument list
–CLI.3.6 [Error code: 220]	File or folder does not exists
–CLI.3.7 [Error code: 219]	Client call failed
–CLI.3.8 [Error code: 218]	Failed creating output directory
–CLI.3.9 [Error code: 217]	Control terminology does not exists
–CLI.3.10 [Error code: 202]	Failed creating output file
–CLI.3.11 [Error code: 201]	Failed processing client task
–CLI.3.12 [Error code: 200]	Failed core process.
–CLI.3.13 [Error code: 199]	File is opened and cannot be written to. Please close the file and restart process
–CLI.3.14 [Error code: 198]	Config for integration library is not found
–CLI.3.15 [Error code: 197]	Enterprise user group is not found
–CLI.3.16 [Error code: 196]	System setup and configuration error. Please update your system
–CLI.3.17 [Error code: 195]	System installation qualification error
–CLI.4.1 [Error code: 215]	Data source is invalid
–CLI.4.2 [Error code: 214]	Argument parsing results must be provided
–CLI.4.3 [Error code: 213]	Standard type is required
–CLI.4.4 [Error code: 212]	Data source is required
–CLI.4.5 [Error code: 211]	Standard version is required
–CLI.4.6 [Error code: 210]	Report file is required
–CLI.4.7 [Error code: 209]	Config is required

Table 3. List of CLI Error Codes

CONCLUSION

The Pinnacle 21 Command Line Interface (CLI) is a handy tool for both Community and Enterprise users and supports all aspects of validation: data validation, define excel template generator, define.xml generator, define validation, and data converter. Current users of Community CLI can easily transition to Enterprise CLI with very few programming changes because the syntax is virtually the same. Because it can be called directly from SAS, the CLI saves precious programming time and can support the automation of an organization's specific workflows. Regardless of how you use the CLI at your organization, the new CLI is an essential tool for SAS programmers that want to automate validation of CDISC data.

RECOMMENDED READING

- <https://www.pharmasug.org/proceedings/2017/HT/PharmaSUG-2017-HT06.pdf>
- <https://www.pinnacle21.com/documentation>

CONTACT INFORMATION

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

Amy Garrett

Pinnacle 21 LLC

agarrett@pinnacle21.com

Brand and product names are trademarks of their respective companies.