

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

CT03

June 6th, 2018

Command Line Interface (CLI)

- › Alternate way to use Pinnacle 21
- › Can be called from SAS
- › Supports automation
- › Increased customization

Notable Changes

Prior releases

- › Task-specific jar files
 - › validator-cli-{version}.jar
 - › data-converter-{version}.jar
 - › define-generator-{version}.jar
- › No support for define.xml validation
- › Syntax:
 - › -parameter=value

Current version

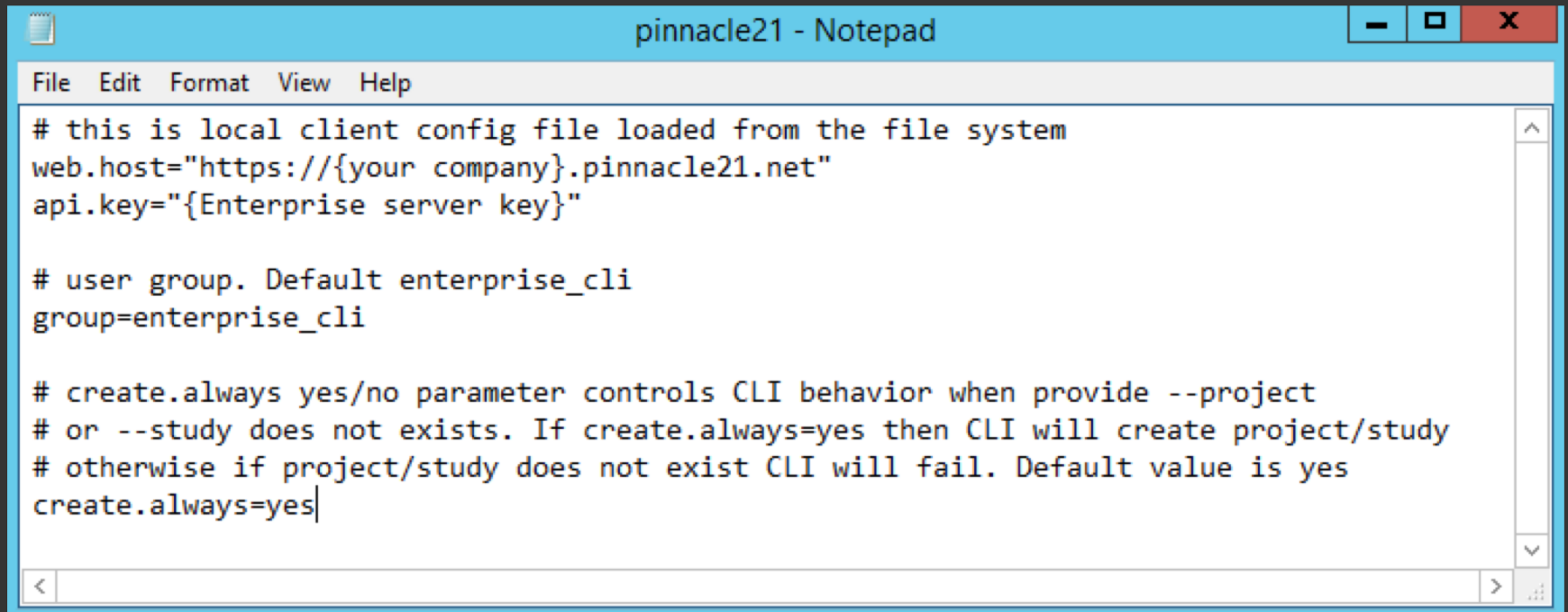
- › Single jar file
 - › p21-client-{version}.jar
- › Support for define.xml validation
- › Syntax:
 - › --parameter=value

Parameter	Valid Values
Standard and Version	
--standard	SDTM, SEND, ADaM
--standard.version	<version>
Validator Source Data	
--source.adam	<File List or Path>
--source.sdtm	<File List or Path>
--source.send	<File List or Path>
--source.define	<File>

Parameter	Valid Values
Terminology and Dictionaries	
--cdisc.ct.adam.version	<version>
--cdisc.ct.sdtm.version	<version>
--cdisc.ct.send.version	<version>
--meddra.version	<version>
--ndf-rt.version	<version>
--snomed.version	<version>
--unii.version	<version>

Parameter	Valid Values
Validator - Report	
--report	<File or Path>
--report.cutoff	<#>
Define.xml Designer	
--source	<File or Path>
--output	<File or Path>

Parameter	Valid Values
--api.key	API Key
--web.host	URL
--group	User group
--project	Project Name
--study	Study Name
--filter	ALL, PMDA, FDA
--datapackage	Data package name
--create.always	Yes No



```
# 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|
```

Example 1: SDTM Validation

Confidentiality

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

Example 2: ADaM Validation

Completeness

```
x
java -jar "&jarpath.\&jarfile."
--standard=adam
--standard.version=1.0
--source.sdtm="aspera://&sdtmpath."
--source.adam="aspera://&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: Define.xml Validation

Enterprise

```
x
java -jar "&jarpath.\&jarfile."
--standard=sdtm
--standard.version=3.2
--project="Project ABC"
--study="S101"
--datapackage="3.2"
--source="&reportpath.\sdtm_define_template.xlsx"
--output="&reportpath.\define.xml"
--report="&reportpath.\define_validation_report.xlsx"
;
```

Return Codes

- › Generated when the --report parameter provided

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

Automation

- › Leverage error codes to kick-off additional workflows
- › Use the CLI to run P21
 - › At the end of dataset creation
 - › After dataset(s) added to a library

Conclusion

- › The CLI supports all aspects of validation
- › Saves time
- › Supports automation
- › Similar syntax, making the transition from Community to Enterprise easy

Questions?

Amy Garrett
agarrett@pinnacle21.com

Aleksey Vinokurov
avinokurov@pinnacle21.net