PharmaSUG 2023 - Paper SD-270

Automating Data Validations with Pinnacle 21 Command Line Interface

Philipp Strigunov, Pinnacle 21

ABSTRACT

Pinnacle 21 Command Line Interface (CLI) allows automation of validation jobs resulting in higher reliability of results and better overall performance of the data preparation process. For example, using the command line tool allows us to integrate Pinnacle 21 software with programming environments including SAS® LSAF, and to work with SAS native SAS7BDAT format. Pinnacle 21 CLI also supports the creation of a Define.xml file. In this paper, we will provide an overview of P21 CLI, explain its syntax parameters, and discuss new features introduced in recent updates. We will also present examples of automating datasets and define.xml validations using shell scripts and SAS ® programming language.

INTRODUCTION

Pinnacle 21 has a versatile data validation toolkit that provides users with two primary options for running validation: a graphical user interface (GUI) or a command line interface (CLI). While the GUI is useful for ad-hoc validations, the CLI is typically utilized for process automation and customization. Common use cases for CLI include triggering validation immediately upon data delivery from a vendor (watched folders), scheduling validations at set times (cron jobs), and calling validations via SAS macros.

In PharmaSUG 2018, the capabilities and syntax of Pinnacle 21 CLI were discussed in detail in the paper How to Automate Validation with Pinnacle 21 Command Line Interface and SAS®. The current paper focuses on the new features introduced in the recent releases of P21 Community and Enterprise CLI (up to versions 1.0.6 and 2.4.0 respectively).

Enterprise CLI (ECLI) has evolved into a separate tool that now supports customizable validations that can digest multiple data formats including native SAS7BDAT. The latest ECLI can be integrated in various systems, from Linux servers to SAS LSAF programming environments.

The recent introduction of REST API opened another layer of opportunities for integrating P21 Enterprise with statistical software. Automated programs can now request metadata for data packages (defines), standards, and terminologies.

GETTING STARTED WITH COMMUNITY CLI

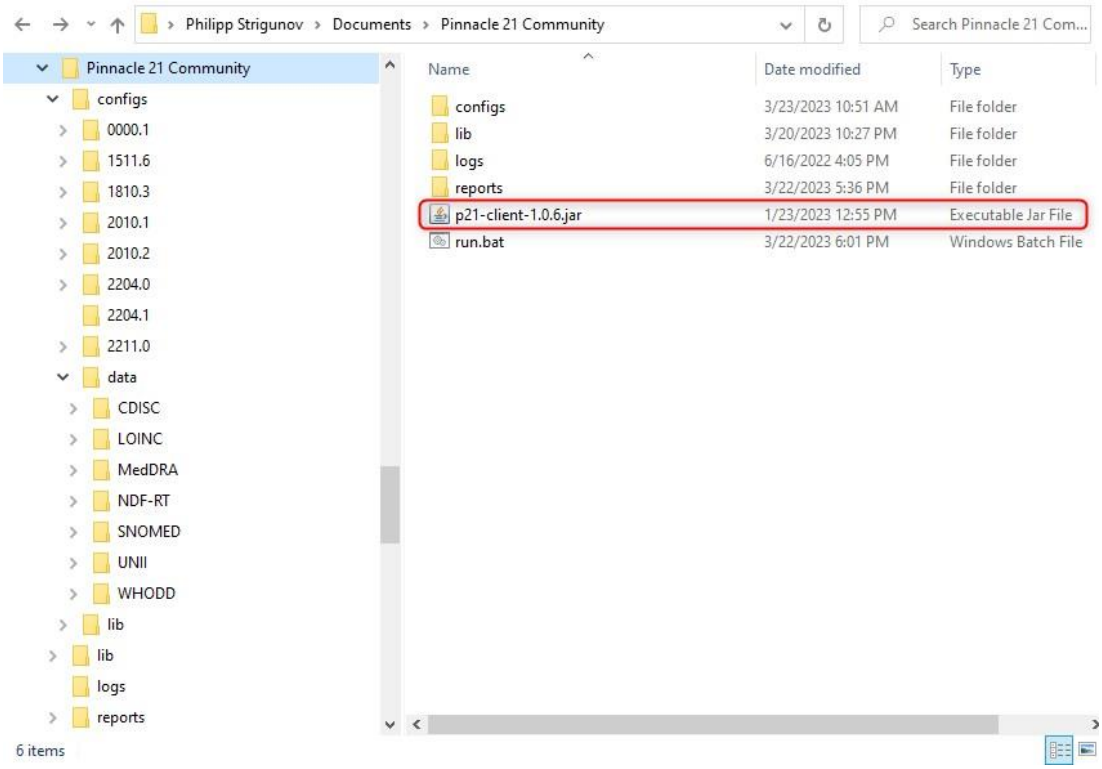
When Community 3.0 was released in 2018, the CLI was completely revamped to make it a first-class citizen. It became a key component with every GUI action (validate data, create spec, generate define, etc.) executed by the CLI. Therefore, you can expect the validation results to be identical between GUI and CLI.

To get started with Pinnacle 21 Community, please visit https://www.pinnacle21.com/downloads and explore the Resources section. There, you can download the latest version of P21 Community app, review documentation, watch webinars, and ask any questions about P21 software on the forum.

INSTALLATION

Community CLI comes bundled with the Pinnacle 21 Community application. The application can be installed in Windows or MacOS. Running CLI requires Java 8 to be installed in the machine. OpenJDK Java Runtime Environment is bundled with the application installer as well and can be used for that purpose.

Community GUI allows users to download all necessary validation engines, configs, and dictionaries. After that, the CLI JAR file can be retrieved from the installation directory and placed next to the Community configs directory. Typical CLI location is illustrated in Display 1. That configuration allows CLI to access the up-to-date metadata locally and run automated validations without using Community GUI.



Display 1. Sample Community CLI Location

COMMUNITY CLI FEATURES AND PARAMETERS

Community CLI mirrors most of the features of the desktop app graphical interface. With that tool, users can validate datasets and Define.xml files and create Define.xml file using datasets and Excel specs. Recent enhancements include the following:

- Users can generate and validate Define-XML 2.1 files
- LOINC dictionary was added in SDTM data validations
- Define.xml generator now supports Analysis Results Metadata

CLI commands are usually executed in batch/shell scripts or SAS macros; however, for exploratory or debugging purposes, a user can run them directly from command prompt or terminal. Each CLI command (request) consists of a set of required and optional CLI parameters. The most common CLI parameters are depicted in Table 1.

Parameter	Valid Values	Description
Validator - Configuration		
--engine.version	“FDA xxxx.y”, “PMDA xxxx.y”, etc	Validation engine to use
--standard	SDTM, ADaM, SEND	Standard used for validation
--standard.version	<version>	Standard version
Validator – Source Data		
--source.adam, --source.sdtm, --source.send, --source.bimo	<Folder or file path>	Absolute or relative path to source ADaM/SDTM/SEND/BIMO data files. Can contain directories or individual files separated by semicolon
--source.define	<File path>	Absolute or relative path to define.xml file

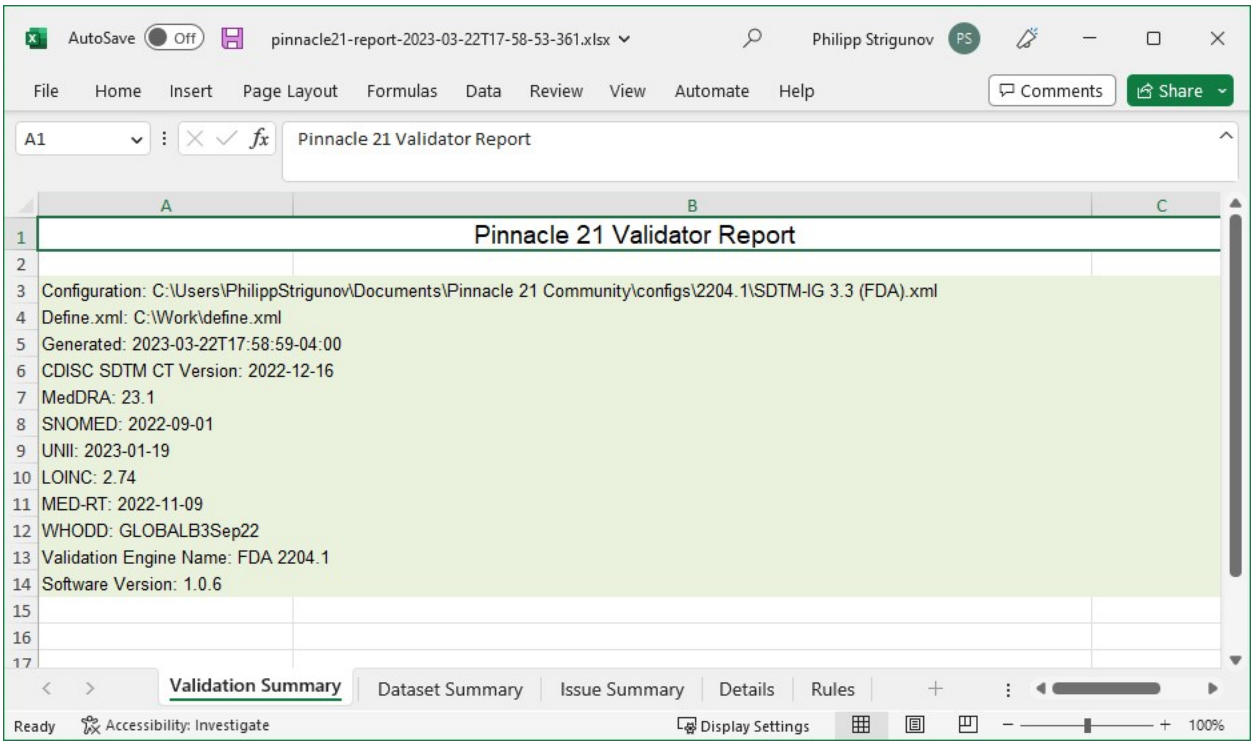
Parameter	Valid Values	Description
--source.encoding	UTF-8, EUC-CN, GB2312, MS936, CP936, GBK, GB18030, Big5	Encoding to be used for reading source data files for validations using NMPA engine
Validator - Terminology		
--cdisc.ct.adam.version	<version>	Version of ADaM controlled terminology
--cdisc.ct.sdtm.version		Version of SDTM controlled terminology
--cdisc.ct.send.version		Version of SEND controlled terminology
--loinc.version		Version of LOINC dictionary in SDTM validations
--meddra.version		Version of MedDRA dictionary in SDTM validations
--med-rt.version		Version of NDF-RT/MED-RT dictionary in SDTM validations
--snomed.version		Version of SNOMED dictionary in SDTM validations
--unii.version		Version of UNII dictionary in SDTM validations
--whodrug.version		Version of WHODrug dictionary in SDTM validations
Validator – Report		
--report	<Folder or file path>	Absolute or relative path to validation report file or folder. If only folder path is provided, CLI will create report file with default name [pinnacle21-report-yyyy-MMddTHH-mm.xlsx]
--report.cutoff	<#>	Number of times a record detail is printed per issue. For no limit specify 0. Default value is 1000.
Define.xml Generator – Create Excel Spec and Convert Spec to Define.xml		
--source	<File path>	Absolute or relative path to Excel Specification
--output	<Folder or file path>	Absolute or relative path to file or folder to place the generated Excel spec or Define.xml.

Table 1. List of Commonly Used CLI Parameters

To illustrate how the CLI parameters are combined, let's start with a command that runs SDTM validation and produces a Community validation report shown in Display 2:

Example 1. CLI SDTM validation command

```
java -jar p21-client-1.0.6.jar ^
    --source.sdtm=C:\Work\data --source.define=C:\Work\define.xml ^
    --engine.version="FDA 2204.1" --standard=sdtm --standard.version=3.3 ^
    --cdisc.ct.sdtm.version=2022-12-16 --meddra.version=23.1 ^
    --whodrug.version=GLOBALB3Sep22 --snomed.version=2022-09-01 ^
    --unii.version=2023-01-19 --ndf-rt.version=2022-11-09 ^
    --report=C:\Work\reports
```



Display 2. SDTM Validation Report Produced with Community CLI

ERROR HANDLING AND PROCESS EXIT CODES

To help users identify any possible issues with the executed command, CLI has a set of specific errors that it can return during the process. The full list of errors and corresponding process exit codes can be found in CLI documentation or by running the following code from the command line:

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

Each error has a corresponding process exit code that can be captured to determine script failure programmatically. For Community CLI, any process exit code higher than 5 signals a task failure. The return code from the last process can be retrieved with `%errorlevel%` in Windows batch scripts and `$?` in shell. Saving exit code to a variable can be useful for triggering consecutive commands. For example, if SDTM validation returns successful exit code, ADaM validation is triggered:

Example 2. Windows Batch Script executing SDTM Validation if ADaM Validation is successful

```
set jarfile=p21-client-1.0.6.jar
set adampath="C:\Work\ADaM"
set sdtmpath="C:\Work\SDTM"
set reportpath="C:\Work\reports"
java -jar %jarfile% --source.adam=%adampath% ^
--engine.version="FDA 2204.1" --standard=adam --standard.version=1.1 ^
--cdisc.ct.adam.version=2022-06-24 --cdisc.ct.sdtm.version=2022-12-16 ^
--report=%reportpath%
if %errorlevel% gtr 5 goto end
java -jar %jarfile% --source.sdtm=%sdtmpath% ^
--engine.version="FDA 2204.1" --standard=sdtm --standard.version=3.3 ^
--cdisc.ct.sdtm.version=2022-12-16 --unii.version=2023-01-19 ^
--med-rt.version=2022-11-09 --loinc.version=2.74 --report=%reportpath%
:end
```

HOW TO INSPECT CLI LOGS

Implementing new CLI commands is not difficult, but if something does not work as intended, check the CLI log file for any errors. CLI log file *p21-client-{version}.log* is stored in the */logs* folder if available; otherwise, it is written right into the same location where the CLI file is placed.

To output additional information, CLI logging can be switched into debug mode. You can switch it 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 …
```

CALLING CLI FROM SAS

Pinnacle 21 CLI is often integrated directly with SAS programming environments. The advantage of that approach is that it eliminates the need for the programmer to exit the programming environment to run Pinnacle 21. CLI calls can be added easily in SAS programs to set a continuous validation process. The following examples will demonstrate some core CLI features by calling the CLI using the `X` statement in SAS.

It is recommended to record common paths and filenames in macro variables so that they can be reused in multiple commands:

```
/* Pinnacle 21 Parameters */
%let jarpath = C:\Users\Documents\Pinnacle 21 Community;
%let jarfile = p21-client-1.0.6.jar;
/* Source Data Parameters */
%let sdtmpath = C:\Work\sdtm\xpt;
%let adampath = C:\Work\adam\xpt;
%let sdtmdefine = C:\Work\define-2.1\define.xml;
%let adamdefine = C:\Work\Study-101\adam\define.xml;
/* Output Parameters */
%let reportpath = C:\Work\reports;
```

Example 3. ADaM Validation using SDTM domains for cross-checks

```
x
java -jar "&jarpath.\&jarfile." ^
--engine.version="FDA 2204.1" ^
--standard=adam ^
--standard.version=1.1 ^
--source.sdtm="&sdtmpath." ^
--source.adam="&adampath." ^
--source.define="&adamdefine." ^
--cdisc.ct.sdtm.version=2022-12-16 ^
--cdisc.ct.adam.version=2022-06-24 ^
--report="&reportpath.\report-adam.xlsx”
;
```

Example 4. Generating Define Excel Template from source datasets

```
x
java -jar "&jarpath.\&jarfile." ^
--engine.version="FDA 2204.1" ^
--standard=sdtm ^
--standard.version=3.4 ^
--source.sdtm="&sdtmpath." ^
--output="&reportpath.\sdtm_define_template.xlsx"
;
```

Example 5. Generate Define.xml from Excel Template

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

Example 6. Define.xml Validation

```
x
java -jar "&jarpath.\&jarfile." ^
--engine.version="FDA 2204.1" ^
--standard=sdtm ^
--standard.version=3.4 ^
--source.define="&reportpath.\define.xml" ^
--cdisc.ct.sdtm.version=2022-12-16 ^
--report="&reportpath.\report-define.xlsx"
;
```

USING CLI WITH P21 ENTERPRISE

Enterprise CLI is used to trigger programmatically jobs that are executed remotely in P21 Enterprise web server. To download the newest Enterprise CLI client and see the documentation, please visit https://help.pinnacle21.net/en/collections/3188814-enterprise-command-line-interface-ecli-api

You don't need to download any additional configuration files, dictionaries, or engines because Enterprise has all metadata with the latest available versions onboard.

ENTERPRISE CLI FEATURES AND ADDITIONAL PARAMETERS

The set of features available in Enterprise CLI depends on the version of Enterprise with which the tool is used. For example, the latest ECLI 2.4.0 supports the capabilities of Enterprise 5.3.1. Both products are frequently updated, but ECLI maintains backwards compatibility with older Enterprise versions. Notable changes brought in latest releases (versions 2.2.0 – 2.4.0) include the following:

- Support for validating ZIP archives and native SAS7BDAT file format.
- Aspera transfers are now supported from within SAS LSAF environments.
- ECLI can export define.xml from P21E Define Designer.
- Dictionaries can be set to always use the latest version, e.g., `--loinc.version="latest"`.
- ECLI runs on full range of Java versions from 8 to 15, official support includes LTS JDK 8 and 11.

ECLI can run in any Windows/Linux environment with Java installed, which expands integration possibilities, in comparison with Community CLI that supports only Windows and macOS.

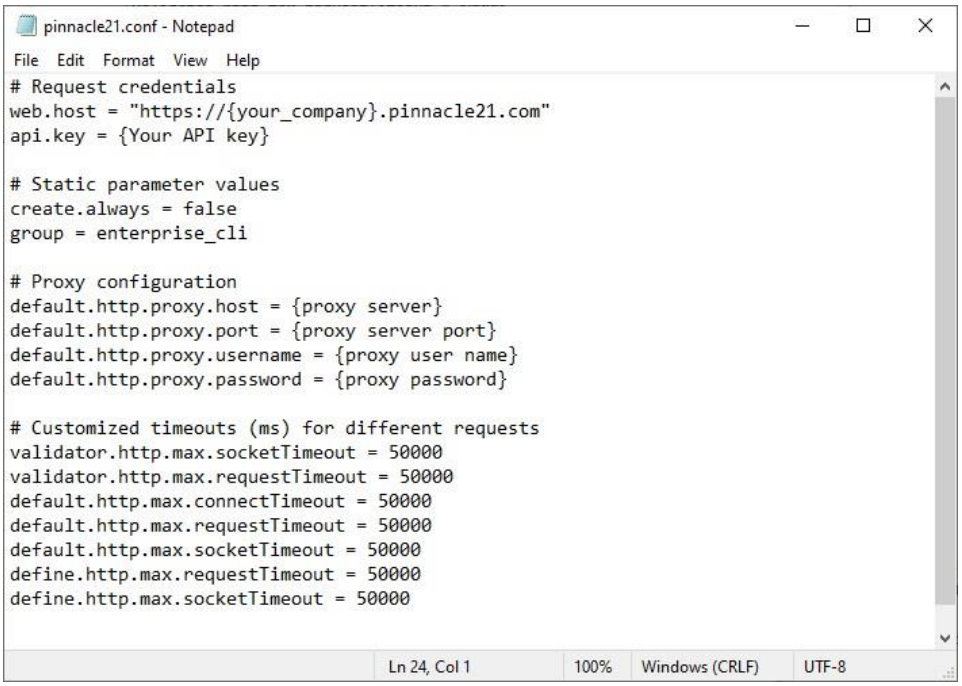
Transition from Community CLI to Enterprise CLI is not difficult because all parameters listed in Table 1 are used in both tools and the commands look similar. There are some additional parameters in Enterprise CLI, as listed in Table 2.

Parameter	Valid Values	Description
Enterprise Objects		
--project	<Project name>	Name of the project to use
--study	<Study name>	Name of the study to use

Parameter	Valid Values	Description
--create.always	true (default), false	Create missing project and/or study based on command line --project/--study parameters if API key has sufficient permissions
--datapackage	<Data package name> (default equals to the standard name)	Name of the data package to use. Will be created if it does not exist. Should be accessible by API key
Data package configuration		
--define.standard	2.0 (default), 2.1	Version of Define-XML standard on the data package
--group	<Group name>	Name of the group to assign to the data package being used
--list.config	true, false (default)	Writes Project, Study, or Data Package configuration to a JSON file path provided in –output
Source data configuration		
--source.comment	<String up to 255 characters long>	Validation comment or note that will appear on the validation history in the UI
--adapter	true, false (default)	Specifies whether a custom adapter format Excel specification is provided in –source
--transfer.target.rate	<Number> (default is specified by the server)	Maximum transfer rate (kbps) for uploading data via Aspera. Set value to 0 for no limit.

Table 2. List of Enterprise CLI Parameters

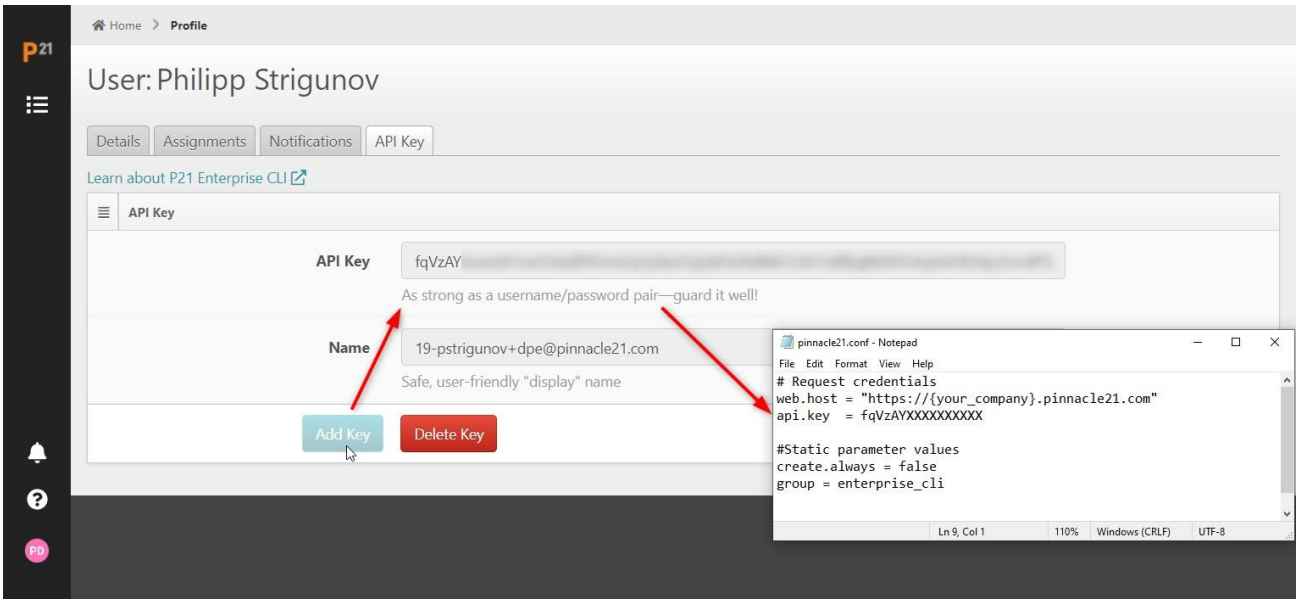
For storing static parameter values, it is recommended to create a *pinnacle21.conf* file in the same directory as the executable CLI JAR file. There you can reference parameters related to connectivity and environment like Enterprise server URL, API key, and proxy server settings. Display 3 shows an example of such a file:



Display 3. Sample pinnacle21.conf File

Please note some important differences between Enterprise and Community CLI tools:

1. To run any ECLI or API task, you need to create an API key in P21 Enterprise. You can use a system-wide API key or create your individual key (more secure and traceable). Personal API key is completely private and has all your user roles and assignments. Add the generated key into the `api.key` parameter in ECLI command or into the *pinnacle21.conf* configuration file, as shown in Display 4. API key roles and assignments required for ECLI tasks are the same as in the Enterprise GUI.
2. Enterprise data validations are performed remotely, so the source data should be sent to the server via IBM Aspera high-performance transport technology. Enterprise CLI makes such uploads easier – you do not need to install any additional Aspera client software because Aspera is built into the CLI tool. You just need to prepend the `aspera://` prefix to source data paths in your CLI commands.
3. All Enterprise CLI requests should use Enterprise Project > Study > Data package structure. Each data package in Enterprise has its own persistent set of dictionary, terminology, and validation engine versions. API key with Data Package Editor role can tweak that set of versions right through ECLI, so it would update the data package before running the validation if needed. For LOINC, SNOMED, UNII, and MED-RT dictionaries, ECLI can configure the data package to always use the latest available dictionary version.



Display 4. Adding Personal API key to pinnacle21.conf

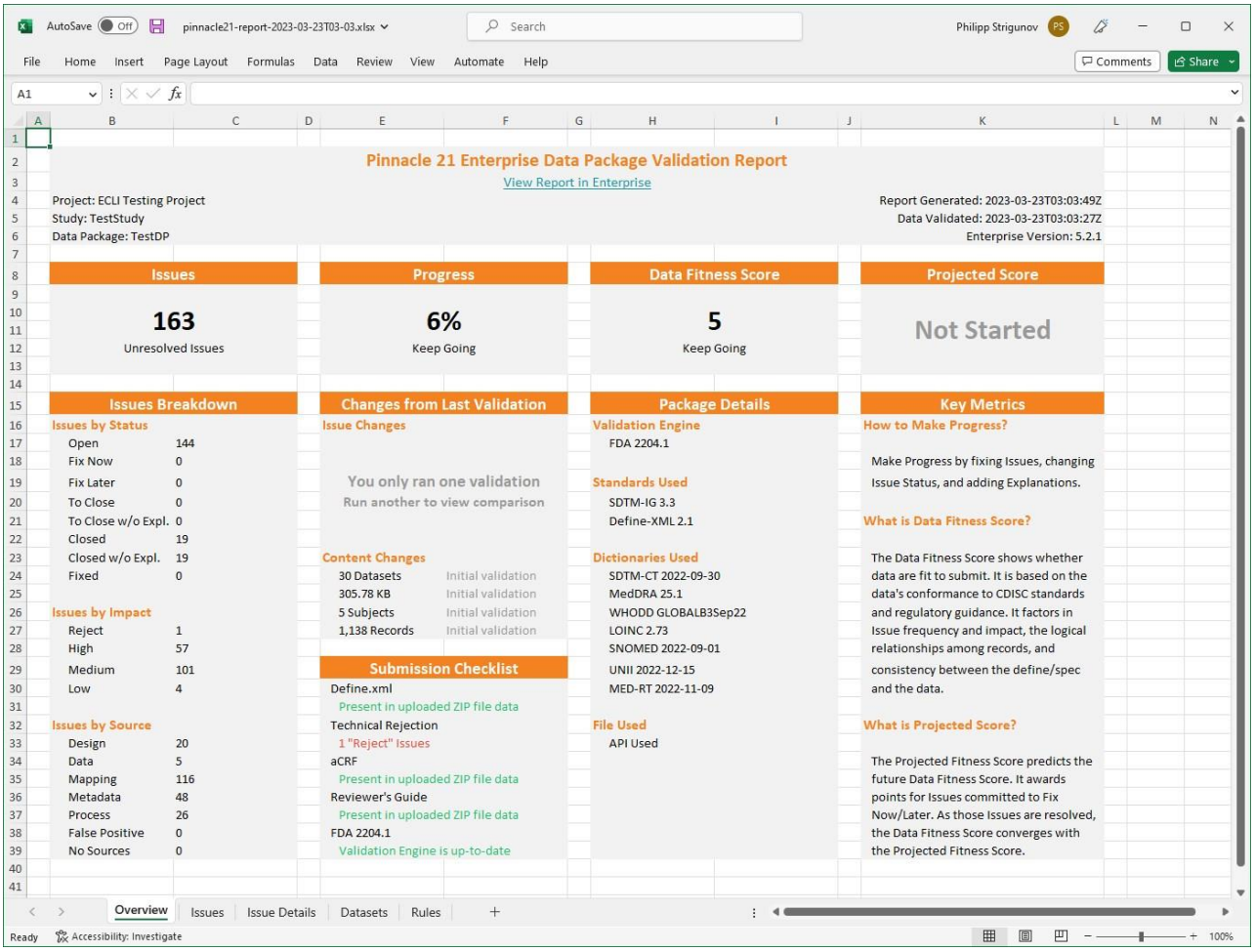
ENTERPRISE CLI EXAMPLES AND USE CASES

Listed below is an example of ECLI SDTM validation command and the produced validation report (Display 5). In this command, ECLI takes Enterprise credentials from *pinnacle21.conf* and creates a new data package:

Example 7. ECLI SDTM validation in a new data package

```
%let jarpath = C:\Users\Documents\cli-package;
%let jarfile = p21-client-2.4.0.jar;
x
java -jar "&jarpath.\&jarfile." ^
--project="ECLI Testing Project" --study=TestStudy --datapackage=TestDP ^
--source.sdtm="aspera://&sdtmpath." ^
--source.define="&sdtmdefine." --define.standard=2.1 ^
```

```
--engine.version="FDA 2204.1" --standard=sdtm --standard.version=3.3 ^
--cdisc.ct.sdtm.version=2022-09-30 --meddra.version=25.1 ^
--whodrug.version=GLOBALB3Sep22 --snomed.version=latest ^
--unii.version=latest --ndf-rt.version=latest --loinc.version=latest ^
--report="&reportpath."
;
```



Display 5. Enterprise Excel Report produced from ECLI SDTM Validation

Example 8. SDTM validation using existing data package attributes

```
x
java -jar "&jarpath.\&jarfile." ^
--project="ECLI Testing Project" --study=TestStudy --datapackage=TestDP ^
--source.sdtm="aspera://&sdtmpath." --source.define="&sdtmdefine." ^
--report="&reportpath."
;
```

Besides dataset validations, Enterprise CLI can also be used to automate tasks for Enterprise Define Designer. For example, it can upload an Excel specification to the data package, generate define.xml, and produce an issue report for that define:

Example 9. ECLI call to generate Define.xml and export its Issue Report from Define Designer

```
x
java -jar "&jarpath.\&jarfile." ^
--project="ECLI Testing Project" --study=TestStudy --datapackage=TestDP ^
--source="&reportpath.\sdtm_define_template.xlsx" ^
--output="&sdtmdefine." ^
--report="&reportpath."
;
```

Example 10. ECLI call to validate Define.xml and export its Issue Report from Define Designer

```
x
java -jar "&jarpath.\&jarfile." ^
--project="ECLI Testing Project" --study=TestStudy --datapackage=TestDP ^
--source.define="&sdtmdefine." ^
--report="&reportpath."
;
```

EXPANDING AUTOMATION FURTHER WITH P21 ENTERPRISE REST API

REST API offers another layer of automation that allows users to integrate their internal systems and workflows with P21 Enterprise. By adding a few lines of code, statistical software can request and receive metadata directly out of your P21E environment to aid in development of new programming code. This metadata can be used to build tools for automation, drive dataset and variable attributes, and even help assign controlled terminology within a SAS program.

REST API is currently available in P21 Enterprise environments version 5.2.0 and later with advanced tier modules enabled.

P21 API works essentially the same way as most websites. When a request is sent from a client to the Enterprise server using a HTTPRequest, the corresponding response is received back over the HTTP/HTTPS protocol using HTTPResponse. Currently, the only supported request method is GET (a request to GET a resource).

API AUTHENTICATION

Basic authentication is typically used in conjunction with HTTPS to provide confidentiality. It requires user credentials to be the Base64 encoding of a string {API Key}:{API Key} sent in a header of the API request. P21 users can retrieve their API key in Enterprise GUI, then encode it in SAS or R using built-in functions. For example, the SAS code below can be used to store the encrypted API key in a macro variable `&apikey`:

```
* Encode the API key in Base64 as key:key (equivalent to: user:pass);
data _null_;
  call symput("apikey", put("&key"||":"||"&key", $base64x500.));
run;
```

ENDPOINT, PATH, AND QUERY PARAMETERS

P21 REST API Request endpoint consists of the P21E environment's URL, API path, and query parameters:

- `/defines/:id` for exporting data package define metadata
- `/standards/:id` for exporting standard metadata
- `/terminologies/:id` for exporting standard codelists and terms

"`:id`" should be replaced with either the data package ID (for /defines/), the standard ID (for /standards/), or the terminology ID (for /terminologies/). An easy way to retrieve the ID is to log into P21

Enterprise, navigate to the metadata that would be the target of the request, and look at the URL in the address bar (see Display 6).

Display 6. Retrieving the Object ID from Enterprise URL

Query parameters include the following:

- `format` – Sets the file format of the requested metadata. Valid options are json (default), pdf, xml, and xlsx.
- `option=byDataset` – If provided, the exported Excel workbook will have one dataset per worksheet.

HTTP STATUS CODES AND ERRORS

After the request is received, the P21 Enterprise server responds with HTTP code status. These codes can range from 100+ to 500+ and generally follow these rules:

- 200+ means the request has succeeded.
- 300+ means the request is redirected to another URL.
- 400+ means an error that originates from the client has occurred. These errors provide context with a message stating the cause. For example, 404 (Not Found) or 400 (Bad Request): Format provided (<format>) is not a valid value. The full list of handled errors can be found in P21 REST API documentation available at https://help.pinnacle21.net/ .
- 500+ means an error that originates from the server has occurred.

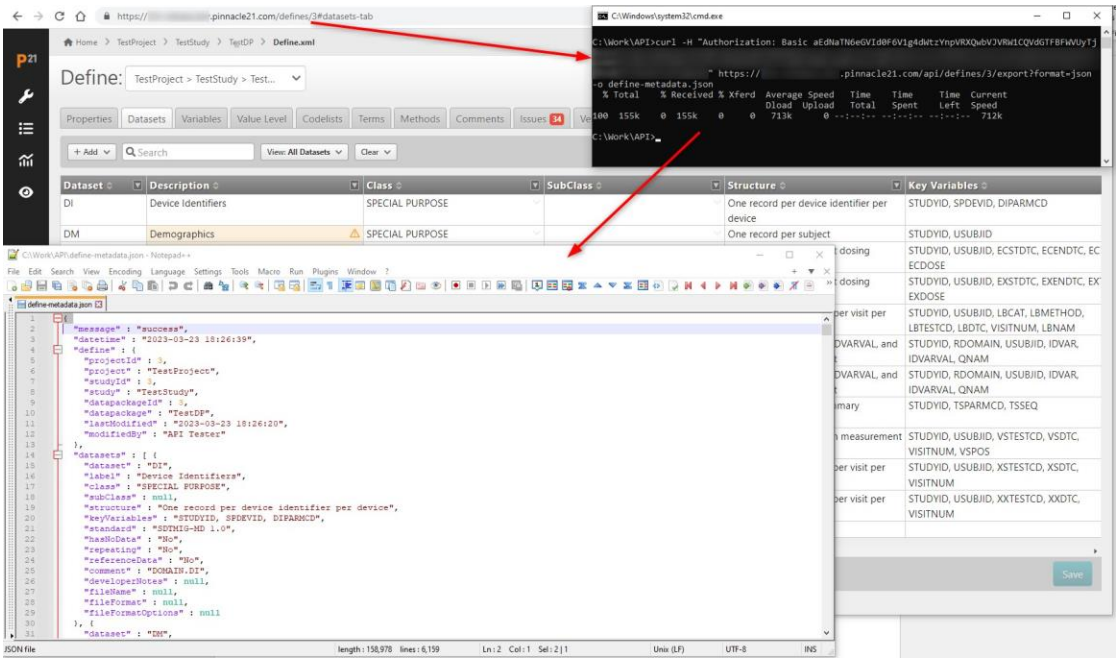
API REQUEST EXAMPLES AND USE CASES

REST API requests can be made with any programming language. The following examples use the popular command line utility called cURL:

Example 11. Request Define.xml Metadata in JSON Format

```
curl -H "Authorization: Basic <Encoded API key>"
https://my-company.pinnacle21.net/api/defines/123/export?format=json
```

This request will return a JSON file if completed successfully. JSON files can contain multiple objects, including arrays of objects, each with a set of key/value pairs. JSON files exported from P21 Enterprise mirror the P21E user interface - one object per tab and a key for each column on the tab. An example of Define 2.1 metadata JSON exported from Enterprise 5.2.1 is illustrated in Display 7.



Display 7. Define.xml Metadata JSON generated with Enterprise REST API

Example 12. Request Define.xml Metadata in Excel Format with one dataset per tab

```
curl -H "Authorization: Basic <Encoded API key>"
https://my-
company.pinnacle21.net/api/defines/123/export?format=xlsx?option=byDataset
```

The following example illustrates exporting standard metadata into SAS and mapping the returned JSON with the SAS JSON libname engine:

Example 13. Requesting standard metadata in SAS using JSON Format

```
* P21E Root URL;
%let root = https://my-company.pinnacle21.com;

* Send API request via PROC HTTP;
filename resp temp;
proc http
  url     = "&root./api/defines/123/export?format=json"
  method  = "GET"
  out     = resp;
  headers
      "Authorization" = "Basic <Base64 API key>";
run;

* Point and read response using LIBNAME JSON;
libname api JSON fileref=resp;

* Clean up filenames and libnames;
filename resp;
libname api;
```

CONCLUSION

The Command Line Interface has been a significant and long-standing feature of Pinnacle 21 products. By taking advantage of CLI tools, you can create study metadata in Define.xml v2.0 or 2.1 format and maintain the CDISC compliance and submission readiness of your data on a continuous basis. Recent updates have allowed CLI to not only match new GUI features of both apps, but also support more programming environments with which the Pinnacle 21 toolkit can be integrated. Regardless of how you use the CLI at your organization, it is an essential tool for SAS programmers that want to automate CDISC data validation.

This paper showed that current users of Community CLI can transition to Enterprise CLI easily with a few programming code changes due to a simple and straightforward command syntax. Cross-platform support allows calling Enterprise CLI directly from SAS environments, including LSAF, to improve data quality control and automate organization workflows.

Recently introduced REST API offers a further automation layer for study, standard, and terminology metadata extraction. This paper described in detail how clinical programmers can utilize Pinnacle 21 command line tools and REST API to simplify daily tasks and create agency compliant deliverables.

RECOMMENDED READING

- Pinnacle 21 Community documentation
  https://www.pinnacle21.com/documentation
- Pinnacle 21 Enterprise CLI documentation
  https://help.pinnacle21.net/en/collections/3188814-enterprise-command-line-interface-ecli-api
- How to Automate Validation with Pinnacle 21 Command Line Interface and SAS®
  https://www.pharmasug.org/proceedings/2018/DS/PharmaSUG-2018-DS20.pdf
- Reading data with the SAS JSON libname engine
  https://blogs.sas.com/content/sasdummy/2016/12/02/json-libname-engine-sas/

CONTACT INFORMATION

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

Philipp Strigunov
Pinnacle 21 by Certara
pstrigunov@pinnacle21.com