

# NIAS

## NSIP Integration As a Service

NAPLAN Student Registration Data Validation

NAPLAN Results Reporting Data — NAPRR

Installation and User Guide v20

## Version history

Version	Date	Notes
V01	4/07/2017	Initial release
V02	10/07/2017	Added Mac user install information
V03	15/08/2017	Added graphql information
V04	09/10/2017	Added v0.99 updates
V05	27/02/2018	Minor updates
V06	17/07/2018	Added Results Reporting Validation/QA reports
V07	07/08/2018	Added updates to NAPVAL validation
V08	29/08/2018	Added Results Reporting Validation reports
V09	11/09/2018	Added Results Reporting Validation reports
V10	04/12/2019	Added to System reports, updates to saving queries
V11	26/02/2019	Added to System reports, updates to acceptable characters in napval.toml file
V12	12/03/2019	Updated redaction
V13	02/05/2019	Updated redaction
V14	06/03/2020	Updated redaction
V15	03/12/2020	Updated redaction
V16	14/04/2020	Splitter functionality, removed GraphQL interface
V17	11/05/2022	Finalised NIAS 2.0.0 documentation of naprrql
V18	16/06/2022	Warning on item report interpretation
V19	26/06/2023	XML redaction per-school
V20	09/10/2023	Miscellaneous updates

## Contents

Section 1 – NSIP Integration As a Service (NIAS) .....	4
1.1 What is NIAS? .....	4
1.2 Installation .....	6
1.3 Updating & Removing NIAS Tools.....	7
1.4 Conditions of Data Download.....	7
Section 2 – Validating NAPLAN Student Registration Data Using NAPVAL.....	8
2.1 Running the Validation .....	8
2.2 NIAS Components .....	15
2.3 NIAS Validation Schemas .....	17
2.4 Conversion of CSV to XML .....	18
Section 3 – Using NAPRRQL .....	20
3.1 NAPRRQL Sample data.....	20
3.2 Loading data files and generating reports .....	23
3.3 Report Functionality .....	<b>Error! Bookmark not defined.</b>
Section 4 – Audit facility NAPLAN Registration .....	24
4.1 Using the audit facility and generating Mismatches report .....	24
Section 5 – NAPLAN Results Reporting Validation and QA Reports .....	26
5.1 Generating Results Validation reports.....	26
5.2 QA Reports.....	26
5.3 System Reports .....	32
5.4 Split Reports.....	38
5.5 Item Printing .....	38
5.6 XML.....	42
5.7 DAC/PNP codes.....	<b>Error! Bookmark not defined.</b>
5.8 Reports available via NAPRRQL user interface .....	<b>Error! Bookmark not defined.</b>
Section 6 Support .....	44
6.1 Support .....	44

## Section 1 – NSIP Integration As a Service (NIAS)

### 1.1 What is NIAS?

NIAS is a suite of open-source components designed to enable as many different users as possible to quickly and easily solve issues of system integration using the Australian SIF Data Model for education. The use of NIAS documented here is for the processing of data around NAPLAN.

The components in the current release perform the following functions:

- ☐ Validation of Student Registration data for NAPLAN
- ☐ Reporting of NAPLAN Results and Reporting data
- ☐ Audit validation between Student Registration data and Results and Reporting data
- ☐ Generation of writing extracts for marking under NAPLAN

These tools are provided for the use of Test Administration Authorities (TAA) and jurisdictions responsible for the upload of NAPLAN Online student registration data to the National Assessment Platform, and the download and processing of NAPLAN results and reporting datasets from the Platform.

The writing extracts functionality is documented separately in the *NIAS Writing Extract User Guide*.

#### NIAS Data Validation (NAPVAL)

The data validation tool allows student registration data files in either .csv or .xml format to be validated to check data format, that mandatory fields in the files are populated, and that the fields are valid against the Registration Data Set specifications.

NAPVAL will also convert .csv files to .xml SIF format once the user is satisfied with the validation.

The user loads the student registration file into the interface, and the validation tool will produce a report of any errors or warnings found, which can be viewed on screen or downloaded. Once the file is validated, users can be confident in uploading the file to the National Assessment Platform for student registration.

#### NAPLAN Results and Reporting (NAPRRQL)

NAPRRQL is a package allowing reporting and browsing of NAPLAN results and reporting data files, for a particular state, sector or schools, provided as a SIF .xml file from the National Assessment Platform.

The user loads the results and reporting file through a command line interface, and NAPRRQL will produce a number of .csv reports for the entire file. It can optionally also break the generated reports down by school, domain, year level, or any combination of the three.

A number of other QA and validation tools can be run using NAPRRQL. The use of these is described in detail further in this document. The table below provides a summary of what is available.

NIAS Validation Tool	Purpose	Refer to
<b>Naprrql.exe –ingest</b>	Ingests results and reporting data. Overwrites existing data.	Section 3

	All other functions will reingest the data by default.	
<b>Naprrql.exe –skipingest</b>	Do not reingest the data: assumes ingest has already been done. Is used in combination with another reporting option.	Section 3
<b>Naprrql.exe –inputFolder ...</b>	Specify the input folder to read the RRD from. If omitted, files are read from the folder “./in”.)	Section 3
<b>Naprrql.exe –report</b>	Generates core reports in csv format.	Section 3, 5.3, 5.4
<b>Naprrql.exe –qa</b>	Generates csv files for validation and checking, in addition to the files generated in <b>report</b> . Recommend this is run before other tools for checking prior to generating reports.	Section 5.2
<b>Naprrql.exe –itemprint</b>	Generates csv file reporting item results for each student against items. (This is excluded from <b>report</b> , because these reports are much more time consuming to generate.)	Section 5.5
<b>Naprrql.exe –xml</b>	Re-extracts redacted xml from Results and reporting dataset	Section 5.6
<b>Naprrql.exe –writingextract</b>	Extract all writing scripts for marking.	See <i>NIAS Writing Extract User Guide</i> in nap-writing-print folder
<b>Naprrql.exe –allReports</b>	Run all reports: includes the reports under <b>report</b> , <b>qa</b> , <b>itemprint</b> , and <b>writingextract</b>	

### Audit Differences (NAPCOMP)

NAPCOMP allows comparison between a NAPLAN Student Registration file (in csv format) and a corresponding Results and Reporting data file received for the same cohort.

The user loads both files into various folders and the executable produces a .txt file which calls out differences where students appear in only one file and not the other. For example, a student may be in the registration data file, but not the results and reporting dataset or vice versa.

None of these components have dependencies and can be run independently at any time.

Please note that instructions and screenshots included in this document are created using sample data which may not reflect realistic scores or results.

## 1.2 Installation

Installation of the NIAS tool loads all of the components as described on previous pages.

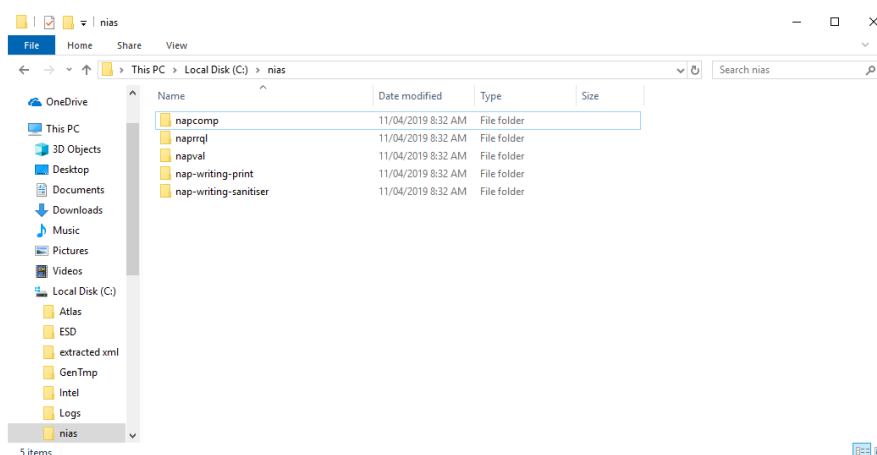
### Pre-requisites

NIAS can be installed on Windows (64-bit), Linux (64-bit), or Macintosh. For the Windows installation you will need a Windows PC with the latest version of either Google Chrome or Mozilla Firefox installed. The Macintosh version works on Chrome, Firefox and Safari.

The NIAS toolset takes up approximately 100MB in hard disk space. You will also need enough hard disk space available to store a copy of the RRD file, and all the reports it generates.

### Installing files

1. Click on the following URL, or enter it in your browser:  
<https://github.com/nsip/nias2/releases>
2. Click on the relevant installation file to download the NIAS tools.
3. **Extract the contents of the go-nias.zip file** to a suitable high-level folder  
e.g.: C:\NIAS



**Note:** Don't nest the folder structure too deeply in Windows; this can have an impact on permissible file name length.

The installation loads 5 folders.

- ☐ **Napval** – enables validation of .csv or .xml files prior to loading into the NAPLAN Online Student Registration Management system
- ☐ **Naprrql** – converts results and reporting dataset into a number of NAPLAN school reports, provides tools for querying results and reporting data, and also contains an audit function
- ☐ **Napcomp**—contains an audit function allowing comparison of a NAPLAN Student Registration Data file against a NAPLAN Results and Reporting file.
- ☐ **Nap-writing-print** and **nap-writing-sanitiser**. These files are further discussed in the *NIAS Writing Extract User Guide*.

Also included in the installation are sample files with which to run the NAPLAN Validation and the NAPLAN Results and Reporting tool. These are further discussed in the NAPVAL and NAPRRQL section of this document.

### 1.3 Updating & Removing NIAS Tools

The tools can be updated by deleting existing folders and downloading a newer version from <https://github.com/nsip/nias2/releases> (Subscribe to this link to be notified of updates to the NIAS tools.)

### 1.4 Conditions of Data Download

Education Services Australia Limited (ESA):

- ☐ complies with the Privacy Act 1988 (Cth) to maintain the privacy of personal information contained within each data extract or report including School Student Summary Report (SSSR) and Individual Student Report (ISR) (Data) while it is stored on the Assessment platform;
- ☐ is unable to control the use of the Data once it has been downloaded from the Assessment platform.

In order to maintain the privacy of the Data after you have downloaded it, prior to accessing and downloading the Data, you confirm and agree that:

- ☐ you are authorised to access and download the Data;
- ☐ your organisation has privacy and security controls in place in order to protect the privacy and security of the Data; and
- ☐ you will take all reasonable steps to ensure that the Data will not be misused, interfered with, lost, modified or disclosed to unauthorised personnel.

If you do not agree to the above conditions, **you must not** access and/or download the Data.

## Section 2 – Validating NAPLAN Student Registration Data Using NAPVAL

Prior to commencing it is assumed that you have a NAPLAN Student Registration data file in the structure contained in the Student Registration Data Specifications v2.0. The file can be .csv or .xml format. Data files in .csv format can be converted to .xml using this tool.

The main changes to the validation process from earlier versions of NIAS (in accordance with updates to the Student Registration Data Specifications) are:

- ☐ Address Fields are no longer required for the platform, so NIAS will report an error if any address fields are populated.
  - The validation will report an error for fields that are defined in SIF but are out of scope of the Registration data set, which now includes addresses.
  - NIAS is intended for XML conversion as an output for the platform, so extra fields not defined in the CSV will not get converted to XML and therefore not uploaded to the platform.
- ☐ Visa codes have been updated

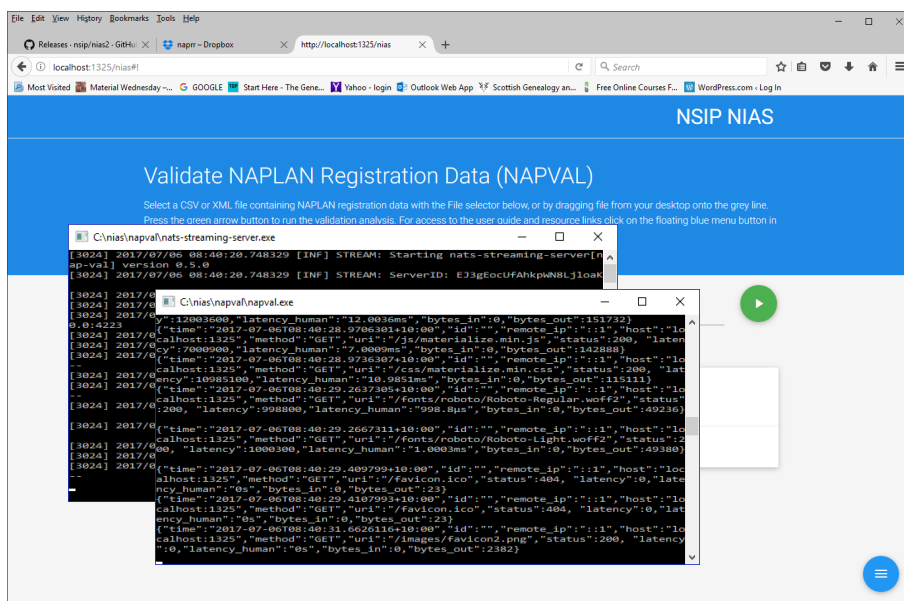
Please refer to v2.0 of the specifications for more detail. A copy is available at <https://github.com/nsip/registration-data-set>

### 2.1 Running the Validation

1. On Windows: To run the validation, navigate to the **napval** subfolder and double-click on **gonapval.bat**. This launches the various components and services of NIAS. On Macintosh and Linux: start **gonapval.sh** from the command line.
2. On launch in Windows, **gonapval.bat** opens a separate command prompt/terminal window for each of the NAPVAL key executables (napval.exe and nats-streaming-server.exe) and launches the NAPVAL web UI using your default web browser (Google Chrome and Mozilla Firefox are supported currently). On Macintosh and Linux, the other executables are launched in the same terminal window; you will need to access the browser to launch NAPVAL yourself, entering the address **localhost:1325**.

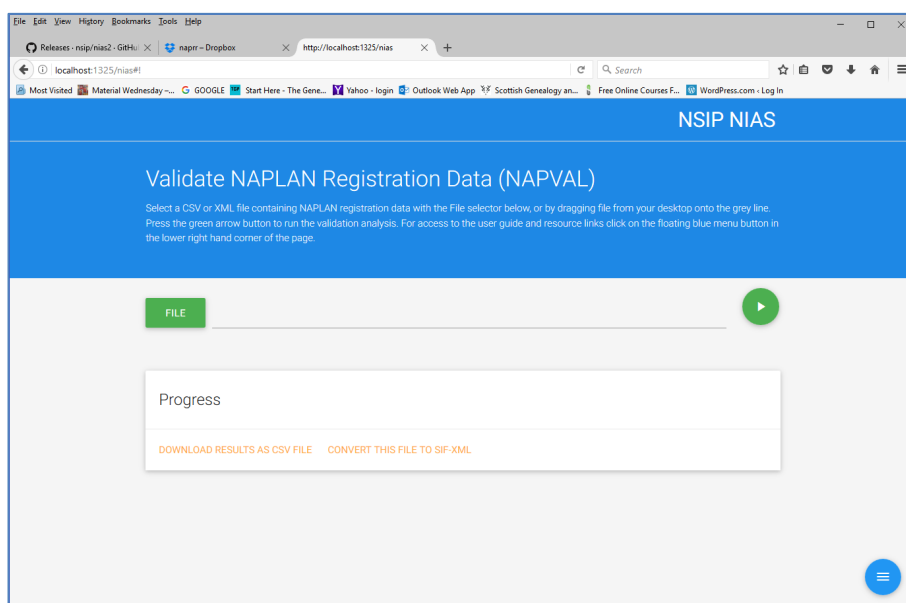
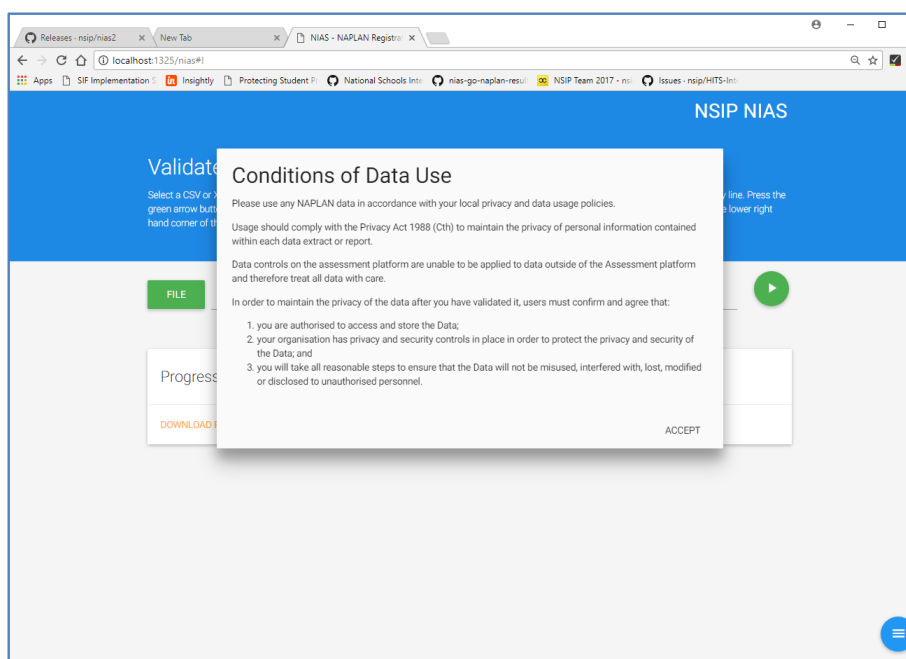
**Note:** *If your default web browser is set to IE or another unsupported browser, the UI may not function correctly.*



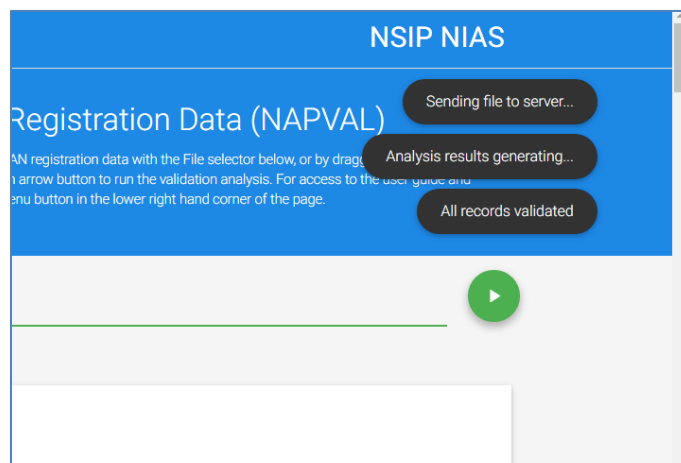


**Important: These windows can be minimised but should not be closed whilst running NAPVAL. Closing these windows terminates that component of the NAPVAL validation tool.**

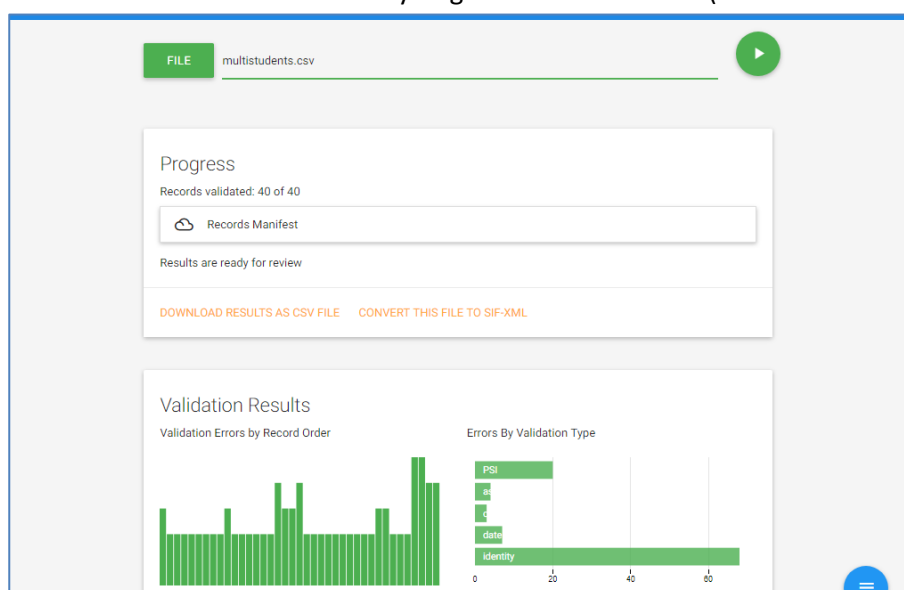
3. The NAPVAL Web UI is displayed. Users are asked to confirm privacy policy on first run – click **ACCEPT** to continue.



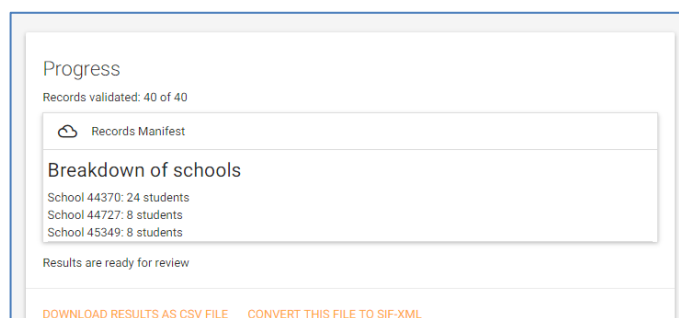
4. Select File and choose a file to be validated from the selection box and click Open, or drop the file in the input location. A sample file, students.csv, is included with the software distribution in the NAPVAL folder. NAPVAL validates files that are either CSV or XML file formats.
5. Click on the play button to commence validation.
6. Progress of the validation is displayed on screen. Do not click on the “Download Results as CSV File” link until all records are validated.



7. NAPVAL performs an initial validation, then lists a summary of the first 100 validation issues on screen.
8. Once file processing is complete, a list of detailed validation results is displayed on screen, broken up into three areas:
  - ☐ Validation errors by record order (graph on left)
  - ☐ Errors by validation type (graph on right)
  - ☐ Error details table – Errors ordered by original file line number (table at bottom)



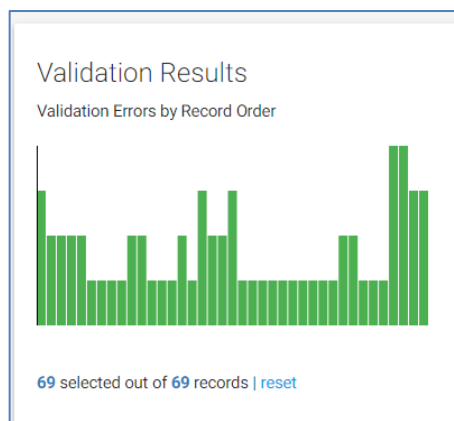
9. There is also a Records Manifest which when expanded will display the breakdown of students records per school. You can use the Manifest to confirm that the uploaded file contains the expected data.



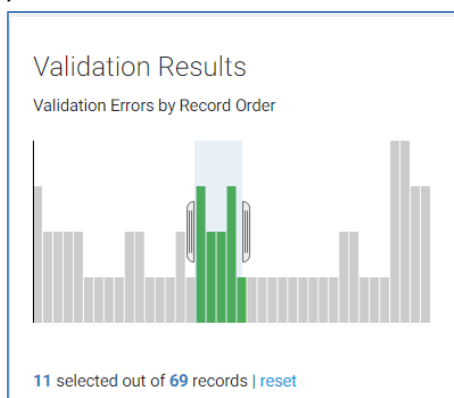
10. To view more detail on the validation errors, use one of the following controls:

- ☐ Validation errors by record order (graph on left) or
- ☐ Errors by validation type (graph on right)

### 10.1 Validation errors by record order

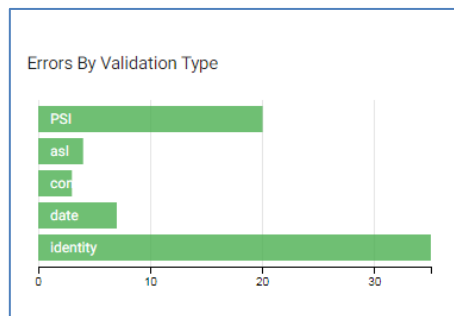


- ☐ This control allows a user the ability to narrow down the validation errors reported based on the record order.
- ☐ Move the mouse cursor over the column graph – a + symbol will appear. Drag this to select a section of the file (selected areas are green). Selecting a section will alter the errors by validation type data displayed and the details below in the errors ordered by original file line number table.
- ☐ As you move the selection window, the system updates the selection of errors displayed based on your selection.

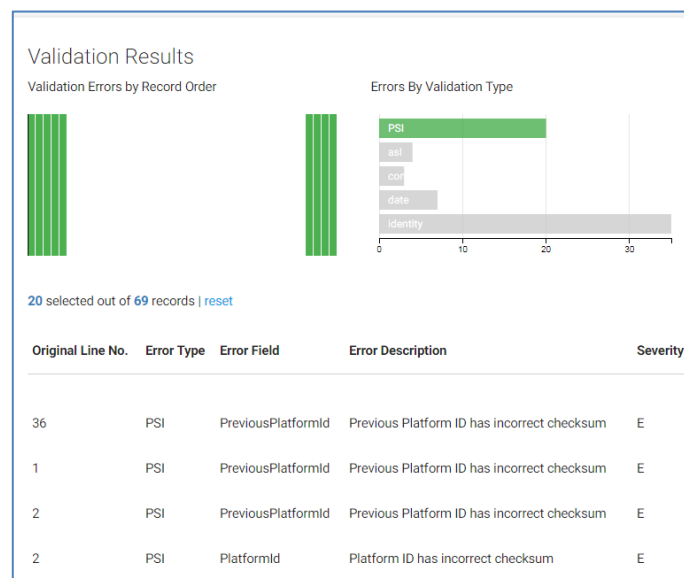


- ☐ To remove any filter on the errors reported, click on **reset**.

## 10.2 Errors by validation type



- This validation control allows the user to focus on a particular validation type (or types) by selecting the appropriate bar/s.
- For example, selecting PSI (= Platform Student Identifier) displays only the PSI-related errors in the detail table and record order graphs.



- You can select more than one validation type to combine results (highlighted in green) by clicking on multiple bars in the graph.

### 10.3 Error details table – Errors ordered by original file line number

69 selected out of 69 records | [reset](#)

Original Line No.	Error Type	Error Field	Error Description	Severity
20	identity	Multiple (see description)	Potential duplicate of record: 5 based on matching: student local id, school asl id, family & given names and birthdate	W
1	PSI	PlatformId	Platform ID has incorrect checksum	E
1	PSI	PreviousPlatformId	Previous Platform ID has incorrect checksum	E
2	PSI	PreviousPlatformId	Previous Platform ID has incorrect checksum	E
2	PSI	PlatformId	Platform ID has incorrect checksum	E
3	PSI	PreviousPlatformId	Previous Platform ID has incorrect checksum	E
3	PSI	PlatformId	Platform ID has incorrect checksum	E
4	PSI	PlatformId	Platform ID has incorrect checksum	E
4	PSI	PreviousPlatformId	Previous Platform ID has incorrect checksum	E
5	PSI	PreviousPlatformId	Previous Platform ID has incorrect checksum	E
5	PSI	PlatformId	Platform ID has incorrect checksum	E
6	identity	Multiple (see description)	Potential duplicate of record: 1 based on matching: student local id, school asl id, family & given names and birthdate	W
7	identity	Multiple (see description)	Potential duplicate of record: 2 based on matching: student local id, school asl id, family & given names and birthdate	W

- ☐ The Error details table displays validation errors detected in the imported file (you can filter it using the control graphs mentioned above).
  - ☐ The table describes the original record number, the type (ASL = Australian Schools List validation, content, identity), the field where the error was detected, the error description, and the error severity (Warning or Error).
11. To export the produced error report, select Download Results as CSV File. You will be able to open or save the file as appropriate. Error reports use the naming convention of OriginalFileName\_error\_report.csv.
  12. After you have reviewed and corrected errors in the source files, you can re-load a new version of the file for review. It is recommended that you refresh the browser if the NAPVAL validation screen is still active. Go back to File, select the revised file, then repeat the file upload steps as detailed above.
  13. To end NAPVAL processing:
    - ☐ On Windows, run **stopnapval.bat** (which will close each terminal window) and close the web browser, or
    - ☐ Close each of the terminal windows by clicking the **x** in the top right corner of each window and close the web browser.
    - ☐ On Macintosh and Linux, run **stopnapval.sh** and close the web browser.

## 2.2 NIAS Components

NIAS contains a number of validation tools. These validation services can be accessed by setting variables in the `napval.toml` configuration file. When you open this file in a text editor, it should look something like this:

```
# Baseline year for DOB checks
TestYear = "2016"

# Validations to invoke
# ValidationRoute = ["schema", "local", "id", "dob", "asl"]
ValidationRoute = ["schema", "id", "dob", "asl"]

# Data match fields for matching across schools
StudentMatch = ["FamilyName", "GivenName", "BirthDate"]

# Legal characters for names. NOTE: this is the input to a regular
expression; if hyphens are permitted, leave - as the final character
LegalNameChars = "A-Za-z '-"

# Webserver port
WebServerPort = "1325"

# Number of validation engines
PoolSize = 4
```

### Configurable variables

NIAS validation variable	Notes
<b>TestYear</b>	Sets the test year during which assessment is running. Used to ensure that students' birth dates align with their test levels.
<b>ValidationRoute</b>	Sets the validations to apply – see table below for details. Currently validations are not order-dependent.
<b>StudentMatch</b>	Fields of the student record to be used for data matching of records between schools during the <i>id</i> check.
<b>LegalNameChars</b>	Sets the legal characters for names. NOTE: this is the input to a regular expression; if hyphens are permitted, leave - as the final character LegalNameChars = "A-Za-z '-" Any instances of backslashes must be doubled, since

	backslashes are a special character in regular expressions: \\
<b>WebServerPort</b>	Sets the web server port. You can edit this if the default conflicts with another service.
<b>NATSPort</b>	Sets the port for the NATS streaming service (used as the bus for messages in the microservice architecture of the software). You can edit this if the default conflicts with another service.

### Validation route values

NIAS validation variable	Notes
<b>asl</b>	Checks that ASL values are correct. Each release of NIAS updates the ASL values to the latest available copy from the Australian Schools List website, <a href="https://asl.acara.edu.au">https://asl.acara.edu.au</a> . (Note: ASL values can be updated by modifying values contained within the asl_schools.csv file contained within the napval/schoolslist folder)
<b>schema</b>	Applies the NAPLAN registration data set validations defined in schemas/core.json.
<b>schema2</b>	Applies dependency validation on NAPLAN registration data set, as defined in schemas/core_parent2.json : ensuring that if one parent 2 value is provided, all of them are provided.
<b>dob</b>	Apply date of birth validation according to the setting of TestYear.
<b>Id</b>	Apply id validation: confirm that every student has a unique LocalId per school, a unique PSI per school, and a unique LocalId, Family Name, Given Name, and Birth Date per school. (The fields used for the



	id check can be changed in configuration). Any collisions within a school are reported. It also detects any students with the same StudentMatch fields between schools.
<b>psi</b>	Apply PSI validation: verifies the check letter for each Platform Student Identifier in the file.
<b>numericvalue</b>	Validates all the numbers with value constraints in the file: currently this only applies to the FTE value, which the service confirms is a decimal number between 0 and 1.
<b>namevalid</b>	Validates student names to confirm that they do not contain extraneous characters, which may be rejected by the Assessment Platform. The legal characters for student names are set in the LegalNameChars configuration parameter, described above.
<b>local</b>	Applies any validations you have set in schemas/local.json. By default this file is not processed, and the default local.json file shipped with nias performs no validation. If you wish to use local validation, rename your local validation file to local.json (more details below).

## 2.3 NIAS Validation Schemas

Validation schemas (in JSON format) are used by NIAS to validate that the input files contain the required mandatory fields and contain valid values. NIAS comes with two validation schemas located in the **schemas** folder and an optional third local validation:

1. Core (*core.json*) – This is the validation schema based on the approved registration data set. It provides the core validation for mandatory/optional fields and valid values. Core validation is always applied when validating file contents. **Do not modify this file.**
2. Core, Parent 2 (*core\_parent2.json*) - This is the validation schema expressing the dependencies between Parent 2 demographic values in registration data: If one such field is present, all of them need to be present. **Do not modify this file.**
3. Local (*local.json*)- This is an optional extra layer of validation which can be applied in addition to the core validation. You can edit this file to include local validations as required.

## local.json

The file local.json can supply an optional extra layer of validation in addition to the validation found in core.json. Typical examples of where extra local validation may be useful include:

1. Making additional fields mandatory (for example a jurisdiction wide identifier)
2. Modifying allowed values (for example removing year level 0)
3. Modifying min/max lengths (for example all jurisdiction identifiers must be 9 characters in length)

The file local.json contains examples of possible local validations which users may seek to implement including a max length of 10 for home group, max length of 36 for jurisdiction id, and an absolute length of 8 for TAAId. It also lists TAAId, JurisdictionId and HomeGroup as mandatory fields.

A suitable JSON or text editor is recommended.

## Enabling local schemas

NIAS users may seek to apply one or more localised schemas (for example SouthAustralia.json, SACatholics.json, SAIndependents.json).

To enable your own additional validation schema:

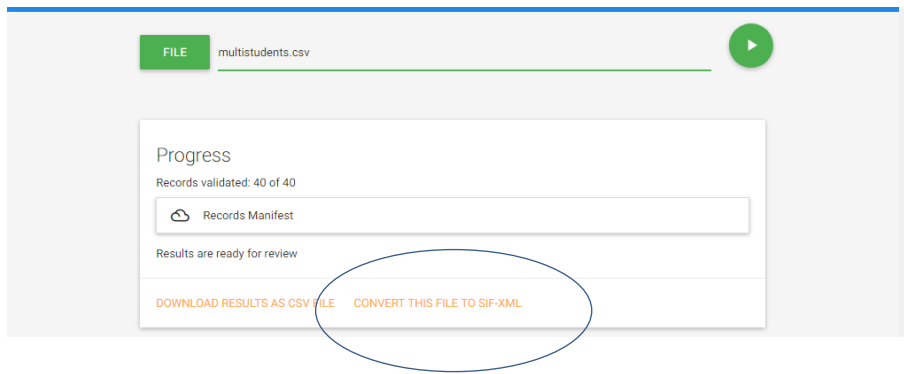
1. In the **napval\schemas** folder, rename local.json to defaultlocal.json
2. Rename your local validation file to local.json
3. Place it in the **schemas** folder
4. In the napval.toml file, uncomment the ValidationRoute with “local” included in the variable arguments, and comment out the default ValidationRoute.
5. Restart NAPVAL and validate your file/s.

Alternatively, you can implement your own validations by directly editing the default local.json file.

## 2.4 Conversion of CSV to XML

Once you are happy with the level of validation, you can convert CSV files to SIF XML format.

1. In the NAPVAL tool select File.
2. Choose the relevant CSV file (this file would typically be the file that has been worked on and validated as required). Select Open.
3. Click the 'Convert this file to XML' link,
4. Save the XML file to an appropriate location. The file will have the same filename as the CSV file but with an XML file extension.
5. You can convert files as many times as you require.



## Section 3 – Using NAPRRQL

NAPRRQL comes with two full results reporting dataset sample extracts, one for 10 schools and 250 students (10schools.xml.zip), and one for 100 schools and 5000 students (100schools.xml.zip), which are ingested on first running the application. In order to run your own data received from the National Assessment Platform, substitute these files with your own zipped XML file; NAPRRQL will read any file suffixed **.xml.zip** in the **\in** folder. Please ensure that your own zipped XML file is not password-protected; you may need to unzip the password-protected file received from the national assessment platform, and re-zip it without a password.

When the application is run, data is streamed from the data file in the **\in** folder and processed onto the reporting sub-system. Report generators are then run to produce .csv extracts of the data in the **\out** folder. This contains aggregate reports at the top level for all report types (score summaries, domain scores, participation and code-frame).

By default, the reports are generated from the input RRD file with no grouping/break down. Optionally, NAPRRQL can also break down reports by school (named by ACARA Id), year level, domain, or any combination of the three. However this only applies to reports that can be meaningfully broken down in this way. Reports which do not break down in this way are ignored; that applies to the codeframe report, for example (systemCodeframe.csv), since it is not specific to a school, year level, or domain.

### 3.1 NAPRRQL Sample data

1. To run NAPRRQL and populate it from the sample file, navigate to the command prompt and to the naprrql folder (eg. C:\nias\naprrql).
2. Run **naprrql.exe -ingest** on Windows, **naprrql --ingest** on Linux and Macintosh. This launches the various components and services of NAPRRQL and begins processing of any data files in the **\in** folder.
3. The application ingests the data in readiness for generating reports in future steps. It displays a progress bar as it does so for each file.

[illegible][illegible]

Once the ingest is complete, to generate NAPLAN Results Reports in .csv format, run **naprrgl.exe -report** on Windows, **naprrgl --report** on Linux and Macintosh. Note that this step will

display progress bars for each type of report.

```

naprrql — Terminal — naprrql --report — 146x38 — ￼%4
2022/03/11 15:07:12 timetrack.go:16: Results data ingest took 6.669s
--- Halting post-ingest.
15:07:13 * ~/Documents/Arbeit/nias2/build/Mac/naprrql[fix/nov2021] $ ./naprrql --report
--- Ingesting results data:

Processing XML File: (in/10schools.xml.zip)
6s [=====] 100%

56970 data-objects parsed

Processing XML File: (in/10schools.xml.zip)
0s [=====] 100%

5190 data-objects parsed

2022/03/11 15:09:04 timetrack.go:16: Results data ingest took 6.569s
--- Data stats:

NAPStudentResponseSet: 23221
NAPCodeFrame: 44
NAPTest: 48
NAPTestlet: 436
SchoolInfo: 110
NAPTestScoreSummary: 1888
StudentPersonal: 5250
NAPTestItem: 4976
NAPEventStudentLink: 26187

--- Running Reports (-QA reports):

Simple reports: 0s [=====] 100%
Student-based reports: 0s [=====] 20%
Codeframe reports: 0s [=====] 100%
Event-based reports: 0s [=====] 24%

```

- This step will generate an **out** folder, which will contain a sub folder for system-reports. If you have enabled and configured the report splitter, this step will then generate an **out\_split** folder. (This functionality is disabled by default.) The **out\_split** folder will also contain a sub folder for system-reports, but that folder will contain sub folders of its own, broken down by school, year level, and/or domain. The reports generated in the **out** and **out\_split** folders are detailed in sections 5.3 and 5.4 of this document.

These reports are available as csv files by viewing them in the **out** folder (and the **out\_split** folder, if enabled). The error messages that appear on screen during the generation of reports are replicated in the file **naprrql.log**.

## 3.2 Loading data files and generating reports

**Important Note:** Because the installation for NAPRRQL includes sample files for testing, it is important that these files are removed first prior to copying the relevant data files to the folders as described below.

The NAPRRQL will process any .xml.zip or .xml file it finds in the folder, which means it will process the sample file along with any files copied in. This also means that multiple files can be loaded into the **\naprrql\in** folder and be processed at the one time. However, those files need to describe distinct students' results. If any XML objects in the files have the same RefIDs (global identifiers), NAPRRQL will no longer abort, as with previous versions of NAPRRQL; but it is no longer predictable which instance of the record will be saved to NIAS. Moreover, those objects will be double counted during ingest, so the progress bars will not give accurate values. That will happen for example if you have a zipped and unzipped instance of the RRD file in the same directory.

The sample files that must be removed are the files 10schools.xml.zip and 100schools.xml.zip contained in the folder: **\naprrql\in**.

Files and folders in the **\naprrql\out** folder will be deleted each time the NAPRRQL tool is run, so it is not necessary to delete sample data from here.

Please refer back to section 3.1 NAPRRQL Sample data for screenshots if required.

1. To run NAPRRQL and populate it using data from the National Assessment Platform, ensure your data file is in **.xml** format and zipped
2. Copy the file to the **\naprrql\in** folder.
3. Navigate to the installation folder in a console or terminal and run **naprrql.exe -ingest** on Windows, **naprrql --ingest** on Linux and Macintosh. This launches the various components and services of NAPRRQL and begins processing of data files in the **\in** folder.
4. The application ingests the data in preparation for reporting. Note that the ingest process only needs to be done once unless the data changes.
5. Once the ingest is complete, to generate NAPLAN Results Reports in .csv format, run **naprrql.exe -report** on Windows, **naprrql --report** on Linux and Macintosh. The reports are divided up into sets which have attributes in common (one row per student, one row per test, one row per item), and run simultaneously. A set of progress bars will indicate how far NAPRRQL has gone in generating the reports: one progress bar for each set of reports.
6. Report generation involves all reports specified in the **config** directory as being activated. NAPRRQL since version 1.2.0 of NIAS (release late 2021) is substantially faster than its predecessors. Nonetheless, processing the file can take a long time; the result set for a jurisdiction will still likely take tens of minutes (though no longer hours), and should be run on a computer with the most memory available. (That is because the speed enhancement involves generating multiple reports simultaneously, which is memory dependent.)
7. Once the report generation is complete, to run the web interface for NAPLAN Results Reporting run **naprrql.exe** on Windows, **naprrql** on Linux and Macintosh. Alternatively, the csv files can be viewed in the **out** folder.
8. **Important: The window can be minimised but should not be closed whilst running NAPRRQL. Closing the window terminates that component of the NAPRRQL reporting tool.**

Note: To help speed the process up, you should disable any reports you will not be using. All reports are activated by default. To disable a report, change its corresponding configuration file in the **config** folder to read “activated = false” instead of “activated = true”. The reports generated by NIAS are described in sections 5.3 and 5.4.

If you wish to read files from a folder other than **\in**, use the option **-inputFolder <folder>**; for example, **naprrql -ingest test** will read files from the folder **test** (inside the **naprrql** folder), instead of **in**.

By default, the **-report** option (and the other options to generate reports, **-qa**, **-itemprint**, **-writingextract**, **-allReports**) will rerun ingest, so strictly speaking the initial ingest is redundant (though it is useful to confirm that the RRD file is uncorrupted.) To skip rerunning ingest with those options, add the option **-skipIngest**.

## Section 4 – Audit facility NAPLAN Registration

The NAPLAN Results Reporting data tool has a facility by which a NAPLAN Registration data file can be compared with an NAPLAN Online Results file to check for records which may be unique to each file. It compares a CSV file of Student Registration records (located in the **Napcomp\in\registration** folder) with the student records in the NAPLAN Online results XML file contained in the **Napcomp\in\results** folder, and detects which students appear only in one or the other file.

The comparison runs in two passes.

- First, records in the two files which have the same Platform Identifier (PSI) are eliminated. (For the comparison to run efficiently, users should endeavour to download from the Student Registration Management system a CSV file containing all student records, and includes their allocated PSIs.)
- Second, all remaining records from the two files are compared according to the fields they have in common, drawn from the following list: Family Name, Given Name, Middle Name, Local Id, PSI, ACARA ID of school, and birth date.
- Be careful to ensure that the birth date in the CSV file is represented in the same format as in the RRD file, as YYYY-MM-DD. If you edit the CSV file in Excel, Excel will tend to autoformat anything that looks like a date in the system default format, which is usually DD/MM/YYYY. The comparison will deem those two dates not to be the same.

### 4.1 Using the audit facility and generating Mismatches report

1. To run the NAPRRQL audit tool, copy the required registration data file (xxx.csv) into the folder: **napcomp\in\registration**
2. The Results (xxx.XML) file should be in the **napcomp\in\results** folder.
3. Navigate to the **napcomp** folder in a console or terminal and run **napcomp.exe**. This launches the various components and services of NAPRRQL and begins the comparison of the csv and xml files.
4. Once the comparison is complete, the mismatches are output to csv files in the **napcomp\out** folder.



- ☐ The file RegisteredButNotInResults.csv contains a listing of the students found to be unique to the NAPLAN Registration file (PSI, user-defined key, and RefId) but not in the Results and Reporting file
- ☐ The file ResultsButNotInRegister.csv contains a listing of all the student records unique to the Results & Reporting file, but not in the Student Registration Results.

## Section 5 – NAPLAN Results Reporting

### 5.1 Reports: General

NIAS generates a number of reports from the results and reporting dataset. Reports are separated into four classes:

- ☐ QA reports, used to validate the integrity of the results and reporting dataset. (We do not expect these to be run outside of ESA.)
- ☐ Core reports, responding to TAA requirements for reporting at the level of students or test scripts.
- ☐ Item-Level reports, including either QA or Core reports at the level of individual item responses. These are generated separately, as they are much more time-consuming to run.
- ☐ Writing Extract reports, intended to enable the generation and validation of writing extracts.

Different command options are used to generate each class of report.

Each report has its own configuration file in the **config** directory. These files give the output file name for each report, the names of each field, and the corresponding internal field that is used to populate it. You normally would not be expected to edit this file, except that you can disable individual reports by setting `activated = false` within them.

### 5.2 QA Reports

1. Navigate to the **naprrql** folder (e.g. C:\NIAS\naprrql)
2. Run **naprrql.exe -qa** on Windows, **naprrql --qa** on Linux and Macintosh. This will generate the QA reports in addition to the normal reports described. This step performs the data ingest step by default; if you want to avoid doing that again, and have already run ingest, run NAPRRQL with the added option **-skipIngest**.
3. This launches the various components and services of NAPRRQL, creates a **qa** folder in the **out** folder and generates the files listed below. The distinction between reports filed under folder qa and reports filed under qa\error\_reports, made in previous versions of NAPRRQL, is no longer made: all reports appear in the same folder.

itemExpectedResponses.csv	
<b>Purpose</b>	<ul style="list-style-type: none"> <li><input type="checkbox"/> Reports one row per student per test</li> <li><input type="checkbox"/> Presently up to 4 Testlets per test</li> <li><input type="checkbox"/> Lists the number of items per testlet that a student is expected to have answered (according to the codeframe), the number of correct/incorrect/not attempted and not presented items in that testlet; and any discrepancies between the number of items expected and referenced in the response</li> <li><input type="checkbox"/> Accounts for any cases where all item responses in the testlet are missing. To highlight these the report injects a dummy "empty testlet report" item response into the testlet which is</li> </ul>

	displayed in the report error. Any students with partial results in their PRD will be found.
<b>What to expect</b>	<input type="checkbox"/> A row per student response per test <input type="checkbox"/> At least one testlet per test populated
<b>Look out for</b>	<input type="checkbox"/> No items in the Not In Path category <input type="checkbox"/> Listings of discrepancies (“Expected Items Not Found, Found Items Not Expected”) to be empty entries (“,”) <input type="checkbox"/> Number of expected items per testlet to equal the correct/incorrect/not attempt/not presented items per testlet

itemWritingPrinting.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports one row of response data per student per test item presented within a NAPLAN test for the Writing domain only <input type="checkbox"/> Detailed information which can be used for ingest into data analysis systems
<b>What to expect</b>	<input type="checkbox"/> Very large report <input type="checkbox"/> Includes a column for each subscore
<b>Look out for</b>	

orphanEvents.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports any NAPStudentEvents whose corresponding school does not have a SchoolInfo object in the data set (no details have been provided about the school, or the school does not exist)
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> Any entry in this report needs to be investigated

orphanScoreSummaries.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports any score summaries whose corresponding school does not have a SchoolInfo object in the data set (no details have been provided about the school, or the school does not exist)
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> Any entry in this report needs to be investigated

orphanStudents.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports any students whose corresponding school does not have a SchoolInfo object in the data set (no details have been provided about the school, or the school does not exist)
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> Any entry in this report needs to be investigated

qaCodeframeCheck.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports all tests, testlets and items referenced in responses that are not included in the Codeframe

<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> Tests / Testlets / TestItems should not be responded to if they are not part of the codeframe: any entry in this report needs to be investigated

qaSchools.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports one row per school, summary of school information as well as data from student registration information and results information <input type="checkbox"/> Includes total registered students and summary of student count by Year level <input type="checkbox"/> Includes total test attempts by test year level and domain <input type="checkbox"/> Includes counts of students by participation status <input type="checkbox"/> Includes count of disruptions
<b>What to expect</b>	<input type="checkbox"/> All schools for the sector should be listed <input type="checkbox"/> Can check response counts against expected students registered per test year level
<b>Look out for</b>	<input type="checkbox"/> Discrepancies between number of students registered at each year level for each test domain; reasonable number of students exempt, absent, experiencing disruptions etc.

qaSystemScoreSummaries.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports a row for every score summary included in the data set, sorted by school, domain and year level
<b>What to expect</b>	<input type="checkbox"/> Use to check expected year levels <input type="checkbox"/> Writing may be included more than once, as score summaries are per test rather than test domain
<b>Look out for</b>	<input type="checkbox"/> Any school without an entry for a domain–year level pair

systemCodeframe.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports all Items against Testlets and Tests in the Codeframe <input type="checkbox"/> Includes Item difficulty, correct answer, item type <input type="checkbox"/> Detailed information which can be used for ingest into data analysis systems
<b>What to expect</b>	<input type="checkbox"/> Large report <input type="checkbox"/> One row for every Item in Codeframe
<b>Look out for</b>	

systemExtraneousCharactersStudents.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports a row for every student whose name contains a character other than a letter, an apostrophe, a hyphen, or space
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> Check for any students whose names may cause SSSR report to crash

systemGuidCheck.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports any GUID that references the wrong kind of object, or doesn't reference any object included in the data set
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> If any data appears, report will show the correct kind of object the GUID should point to. Any entry in this report needs to be investigated.

systemMissingTestlets.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports a row for every student response set with a participation status of P, which contain less testlets (according to the ParallelTest field of the student response set object) than are expected for the given test level and test domain. (For 2022, these are: three for Reading and Spelling; three for Numeracy Yrs 3 and 5; four for Numeracy Yrs 7 and 9; one for Grammar & Punctuation and for Writing.)
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> Check for any students that have not been presented the full count of testlets expected. Check the test disruptions and total lapsed time for the test event, as potential explanations.

systemObjectFrequency.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports for every student the number of events they have recorded, the number of events with participation status P/R/S (Participated/Refused/Sanctioned Abandonment)—which are expected to have produced responses; and the number of responses <input type="checkbox"/> Also includes P/R/S events without responses, and responses without P/R/S events
<b>What to expect</b>	<input type="checkbox"/> A row for every unique student in the data set, independent of school
<b>Look out for</b>	<input type="checkbox"/> Check for discrepancies the number of events with participation status P/R/S and the number of responses <input type="checkbox"/> Check for any instances of a student participating in more than five distinct events (as identified by test level plus test domain)

systemParticipationCodeImpacts.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports tests for which the response contents are unexpected based on the participation code
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> "Adaptive pathway without student undertaking test" if: <ul style="list-style-type: none"> <li>○ participation code = S, P</li> <li>○ PathTakenForDomain or ParallelTest are not empty</li> </ul> <input type="checkbox"/> "Scored test with status other than P or R" if: <ul style="list-style-type: none"> <li>○ participation code ≠ P, R</li> <li>○ RawScore or ScaledScore is not empty</li> </ul> <input type="checkbox"/> "Non-zero score with status of R" if:

	<ul style="list-style-type: none"> <li>○ participation code = R</li> <li>○ RawScore is not empty</li> <li>○ RawScore ≠ 0</li> <li><input type="checkbox"/> “Unscored test with status of P or R” <ul style="list-style-type: none"> <li>○ participation code = P, R</li> <li>○ RawScore or ScaledScore is empty</li> </ul> </li> </ul>
--	--

systemParticipationCodeItemImpacts.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports items for which the item responses are unexpected based on the participation code, focusing on score at test level, testlet level and item level
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> “Response captured without student writing test” if: <ul style="list-style-type: none"> <li>○ participation code ≠ S, P</li> <li>○ Response is not empty</li> </ul> <input type="checkbox"/> Scored test with status other than P or R if: <ul style="list-style-type: none"> <li>○ participation code ≠ P, R</li> <li>○ TestletScore, ItemScore or Subscores are not empty</li> </ul> <input type="checkbox"/> Non-zero Scored test with status of R if: <ul style="list-style-type: none"> <li>○ Participation code = R</li> <li>○ TestletScore, ItemScore or Subscores is not empty</li> <li>○ TestletScore, ItemScore or Subscores ≠ 0</li> </ul> <input type="checkbox"/> Unscored test with status of P or R <ul style="list-style-type: none"> <li>○ Participation code = P, R</li> <li>○ ItemScore is empty</li> </ul> <input type="checkbox"/> Unscored writing test with status of P <ul style="list-style-type: none"> <li>○ ParticipationCode = P</li> <li>○ TestDomain = Writing</li> <li>○ Subscore is empty</li> </ul>

systemResponses.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports one row for every student per school registered for a test domain, along with information on their response if available <input type="checkbox"/> Includes participation code, path taken for domain and raw score <input type="checkbox"/> Detailed information which can be used for ingest into data analysis systems
<b>What to expect</b>	<input type="checkbox"/> Very large report <input type="checkbox"/> Will include a row for every student registration for all tests, but not all rows will contain responses
<b>Look out for</b>	

systemRubricSubscoreMatches.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports item/response pairs in which the rubric types and subscore types do not match
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty

<b>Look out for</b>	<input type="checkbox"/> Where there are Expected Rubrics Not Used, which are subscores that do not include a rubric in the list of 10 expected rubric names <input type="checkbox"/> Where there are Used Rubrics Not Expected, which are subscores with rubrics that are not in the list of 10 expected rubric names <input type="checkbox"/> Where there are Subscores Not Defined, which are subscores with no corresponding rubric <input type="checkbox"/> Where there are Rubrics Not Scored, which are rubrics with no corresponding subscore in the item response
---------------------	---

systemStudentEventAcaraldDiscrepancies.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports a row for every test administered at a different school from where the student was enrolled
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> Check for any students whose school of enrolment and school administering the test conflict. If a student has been registered against two schools, the first school of registration will be used as the control.

systemTestAttempts.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports all sanctioned abandonments for students in schools for sector
<b>What to expect</b>	<input type="checkbox"/> Small number of rows
<b>Look out for</b>	

systemTestCompleteness.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports for each test per year level per school in sector counts for Present and Sanctioned Abandonment, and discrepancies between attempts (events with status P or R) and responses
<b>What to expect</b>	<input type="checkbox"/> Row per school per domain per test level
<b>Look out for</b>	<input type="checkbox"/> List of attempts with no response should be empty for all rows <input type="checkbox"/> List of responses with no attempts should be empty for all rows <input type="checkbox"/> P_Attempts + S_Attempts + R_Attempts should add up to Responses

systemTestIncidents.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports all non-empty test disruptions by student, school and domain
<b>What to expect</b>	<input type="checkbox"/> Small number of rows for disruptions impacting individual students <input type="checkbox"/> Larger number of rows for disruptions impacting cohort of students
<b>Look out for</b>	<input type="checkbox"/> Schools with high proportion of disruption

systemTestTypeImpacts.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports tests for which the response contents are unexpected based on the test domain
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> “Writing test with adaptive structure” if <ul style="list-style-type: none"> <li>○ TestDomain = Writing</li> <li>○ PathTakenForDomain or ParallelTest is not empty</li> </ul> <input type="checkbox"/> “Non-Writing test with non-adaptive structure” if <ul style="list-style-type: none"> <li>○ TestDomain ≠ Writing</li> <li>○ PathTakenForDomain or ParallelTest is empty</li> </ul>

systemTestTypeItemImpacts.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports items for which the item responses are unexpected based on the test domain, focusing on score at test level, testlet level and item level
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> “No subscores for Writing Test” if <ul style="list-style-type: none"> <li>○ TestDomain = Writing</li> <li>○ Subscores is empty</li> </ul> <input type="checkbox"/> Subscores for non-writing test if <ul style="list-style-type: none"> <li>○ TestDomain ≠ Writing</li> <li>○ Subscores is not empty</li> </ul>

systemStudentTestYearLevelDiscrepancies.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports a row for every test administered at a different test level from the year level that the student was enrolled in
<b>What to expect</b>	<input type="checkbox"/> Ideally this report should be empty
<b>Look out for</b>	<input type="checkbox"/> Check for any students whose year level conflicts with the test level they were administered.

### 5.3 System Reports

The following are core reports extracted from the RRD file, generated in the `system_reports` folder as a result of using the **-report** option in NIAS. As with QA reports, if you want to avoid reingesting the RRD file, add the **-skippingest** option.

In the following,

- ☐ reports prefixed with `act` are specific to ACT,
- ☐ reports prefixed with `nsw` are specific to NSW,
- ☐ reports prefixed with `qcec` are specific to the QCEC,
- ☐ reports prefixed with `qcaa` and `qld` are specific to the QCAA,
- ☐ reports prefixed with `sa` are specific to South Australia.
- ☐ Reports specific to a jurisdiction are coloured in orange.

actSystemDomainScores.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports raw and scaled scores for each student, as well as proficiency level, year level, domain, parallel path, and participation



<b>What to expect</b>	<input type="checkbox"/> One row per student per test domain
<b>Look out for</b>	Proficiency level (updated in 2023)

compareItemWriting.csv	
<b>Purpose</b>	<input type="checkbox"/> Used to compare successive writing extract instances (using <b>csvdiff</b> mode) <input type="checkbox"/> Reports student identifiers, participation code, word count, start time, and item response
<b>What to expect</b>	<input type="checkbox"/> One row per student for each student that has done a writing test in the current writing extract.
<b>Look out for</b>	<input type="checkbox"/> Participation statuses that have changed from P between extracts: any such participation statuses indicate students whose results need to be re-marked. <input type="checkbox"/> Any changes in word count or text of writing responses.

compareRRDtests.csv	
<b>Purpose</b>	<input type="checkbox"/> Used to compare overall status of NAPLAN testing over successive writing extract instances (using <b>csvdiff</b> mode) <input type="checkbox"/> Reports student identifiers, test IDs, participation code, start time, and parallel paths
<b>What to expect</b>	<input type="checkbox"/> One row per student for each student for each test registered against in the current writing extract.
<b>Look out for</b>	<input type="checkbox"/> Any changes in participation statuses or parallel paths between instances.

isrPrinting.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports for student including local ID, name, scores for each domain, mean scores <input type="checkbox"/> Can be used to facilitate merge with paper records to support printing of hardcopy ISRs.
<b>What to expect</b>	<input type="checkbox"/> A record for each student
<b>Look out for</b>	

isrPrintingExpanded	
<b>Purpose</b>	<input type="checkbox"/> Reports for student including local ID, name, and, for each domain: scaled scores for the student, scaled score standard deviations for the student, mean scores (across the school), pathways for test responses, and full demographic information for each student. Report produced at the request of CEO WA. <input type="checkbox"/> Can be used to facilitate merge with paper records to support printing of hardcopy ISRs.
<b>What to expect</b>	<input type="checkbox"/> A record for each student
<b>Look out for</b>	

nswItemDescriptors.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports the item descriptor for each item in the codeframe
<b>What to expect</b>	

<b>Look out for</b>	
---------------------	--

<b>nswPrint.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Reports the domain scores, participation status, and DAC/PNP codes for each test sat by a student, following NSW TAA format requirements
<b>What to expect</b>	<input type="checkbox"/> A record for each student, including all test domain responses
<b>Look out for</b>	

<b>nswWritingPearsonY3.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Output the writing results of year 3 students, following the fixed-width format prescribed by Pearson for processing.
<b>What to expect</b>	<input type="checkbox"/> A record for each student, with fixed width fields rather than comma-delimited fields.
<b>Look out for</b>	<input type="checkbox"/> There should be no writing results included in the report, since none are present in the RRD file: the file is to be used as a template to be populated by merging the writing results in, out of band.

<b>nswWritingPearsonY5.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Output the writing results of year 5 students, following the fixed-width format prescribed by Pearson for processing
<b>What to expect</b>	<input type="checkbox"/> A record for each student, with fixed width fields rather than comma-delimited fields.
<b>Look out for</b>	<input type="checkbox"/> There should be no writing results included in the report, since none are present in the RRD file: the file is to be used as a template to be populated by merging the writing results in, out of band.

<b>nswWritingPearsonY7.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Output the writing results of year 7 students, following the fixed-width format prescribed by Pearson for processing
<b>What to expect</b>	<input type="checkbox"/> A record for each student, with fixed width fields rather than comma-delimited fields.
<b>Look out for</b>	<input type="checkbox"/> There should be no writing results included in the report, since none are present in the RRD file: the file is to be used as a template to be populated by merging the writing results in, out of band.

<b>nswWritingPearsonY9.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Output the writing results of year 9 students, following the fixed-width format prescribed by Pearson for processing
<b>What to expect</b>	<input type="checkbox"/> A record for each student, with fixed width fields rather than comma-delimited fields.

<b>Look out for</b>	<input type="checkbox"/> There should be no writing results included in the report, since none are present in the RRD file: the file is to be used as a template to be populated by merging the writing results in, out of band.
---------------------	--

<b>qcaa_napo_event_student_link.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each NapeventStudentLink object
<b>Look out for</b>	

<b>qcaa_napo_items.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each NAPTestItem object
<b>Look out for</b>	

<b>qcaa_napo_schools.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each SchoolInfo object
<b>Look out for</b>	

<b>qcaa_napo_student_response_set.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each NAPStudentReponseSet (the top level scoring information)
<b>Look out for</b>	

<b>qcaa_napo_students.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each StudentPersonal object
<b>Look out for</b>	

<b>qcaa_napo_testlet_items.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each mapping of a testlet to a test item in NAPTestlet
<b>Look out for</b>	

<b>qcaa_napo_testlets.csv</b>	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each NAPTestlet object
<b>Look out for</b>	

qcaa_napo_tests.csv	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each NAPTest object
<b>Look out for</b>	

qcaa_napo_writing_rubric.csv	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each subscore received against a rubric in a NAPStudentResponseSet object for writing
<b>Look out for</b>	

qcaa_test_score_summary.csv	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each NAPTestScoreSummary object
<b>Look out for</b>	

qldStudent.csv	
<b>Purpose</b>	<input type="checkbox"/> Basic registration information for each student by school in TAA, following QLD TAA format requirements
<b>What to expect</b>	
<b>Look out for</b>	

qldStudentScore.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports Student ID Domain, Year level, raw and Scaled scores, Band Participation status for each student, following QLD TAA format requirements
<b>What to expect</b>	
<b>Look out for</b>	

qldTestData.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports item information per year level and domain, includes sequence and item type, following QLD TAA format requirements
<b>What to expect</b>	
<b>Look out for</b>	

saStudentParticipation.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports on student participation, following SA TAA format requirements
<b>What to expect</b>	<input type="checkbox"/> A record for each student, including home schooling status, test attempt status and student demographics.
<b>Look out for</b>	

studentProficiency.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports student proficiency levels reached for each test.
<b>What to expect</b>	<input type="checkbox"/> A record for each test result, with demographic information for the student, and proficiency values.
<b>Look out for</b>	<input type="checkbox"/> The proficiency values will only be populated alongside the scaled scores, in the final release of the RRD, and should be drawn from a very limited set of values.

systemCodeframe.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports for each year level and domain, Testlet information including Name, Location, Node, Maximum Score and Items, including difficulty Maximum score, difficulty and Item Type
<b>What to expect</b>	
<b>Look out for</b>	

systemDomainScores.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports raw and scaled scores for each student, year level and domain
<b>What to expect</b>	<input type="checkbox"/> One row per student per test domain
<b>Look out for</b>	

systemObjectsCount.csv	
<b>Purpose</b>	<input type="checkbox"/> Summary report showing counts for: schools, students, test events, student responses, tests, testlets, test items, codeframes & score summaries
<b>What to expect</b>	
<b>Look out for</b>	

systemPNPEvents.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports on all students with at least one PNP (DAC) code for an event.
<b>What to expect</b>	<input type="checkbox"/> One row for each test event which has at least one associated PNP code.
<b>Look out for</b>	

systemParticipation.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports participation codes for each student in each school by domain
<b>What to expect</b>	
<b>Look out for</b>	

systemSchools.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports school registration details, including school type, sector, principal name and websites

<b>What to expect</b>	
<b>Look out for</b>	<input type="checkbox"/> Student count will only be populated if it has been submitted in Registration

	systemScoreSummaries.csv
<b>Purpose</b>	<input type="checkbox"/> Reports for each school the Local Test Name and averages for School, Jurisdiction and National
<b>What to expect</b>	
<b>Look out for</b>	<input type="checkbox"/> Writing may be included more than once, as score summaries are per test rather than test domain <input type="checkbox"/> Any school without an entry for a domain–year level pair

## 5.4 Split Reports

By default, NAPRRQL generates only reports covering the entirety of the data it has ingested. However, NAPRRQL also has functionality that can break down reports according to school, domain, and/or year level, according to how it is configured, and output to a separate folder (by default **out\_split**). This functionality replaces the **out\school\_reports** reports generated in previous versions of NAPRRQL, which only allowed breakdown according to school.

In order to ensure that reports are broken down, edit the file **config\_split\config.toml**, to read `Enabled = true` under the `[Splitting]` heading, instead of `Enabled = false`. The line `Schema = ["School", "YrLevel", "Domain"]` indicates that, wherever possible, reports will be broken down by all three of School, Year Level, and Domain: so any one report file in the broken down version will only contain results applicable to a single school, a single year level, and a single domain. Reports will be filed in a subfolder relevant to its contents. (For example, the subfolder **out\_split\21212\3\Numeracy** will contain reports that include only results for Year 3 Numeracy tests in the school with ACARA ID 21212.) If you require fewer levels of breakdown, remove one or two of the options.

If a report cannot be broken down by domain, because it contains columns relating to multiple domains, its domain folder will be named `AllDomains`.

Splitting functionality only applies to those reports whose configuration file (under `config`) includes as its first three fields `"School" = "CalculatedFields.SchoolId"`, `"YrLevel" = "CalculatedFields.YrLevel"`, and `"Domain" = "CalculatedFields.Domain"`. If the config file does not have those fields at the start, it cannot be processed by the report splitter, and it will be ignored. For example, the report configuration file `config\ActSystemDomainScores.toml` does include those fields, so `system_reports\ActSystemDomainScores.csv` can be split. The report configuration file `config\SystemCodeFrame.toml` does include those fields, so `system_reports\SystemCodeFrame.csv` cannot be split.

## 5.5 Item Level Reports

Certain reports have one row for each item response of a student, rather than one row per student or per test script. This makes those reports much larger and much more time consuming to generate. For that reason, they are only optionally generated.

To generate item level reports, run **naprrql.exe -itemprint** on Windows, **naprrql --itemprint** on Linux and Macintosh. This generates report corresponding to all individual item responses for all domains. If you run **naprrql.exe -allReports** on Windows, **naprrql --allReports** on Linux and Macintosh, all reports will be generated, incorporating item level reports as well as QA and core reports.

This creates the following reports under the **out** folder, including both QA and core reports.

qa/qaStudentResultsCheck.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports the number of times an item appears in a test response that is not anticipated in the codeframe
<b>What to expect</b>	<input type="checkbox"/> This report should be empty
<b>Look out for</b>	<input type="checkbox"/> Any rows present in the report should be investigated, as they point to a discrepancy in structure between the codeframe and the administration of test items

qa/systemItemCounts.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports the number of times an item is in a response at all
<b>What to expect</b>	<input type="checkbox"/> A row for every item in the codeframe
<b>Look out for</b>	<input type="checkbox"/> High counts for substitute items <input type="checkbox"/> Zero counts for any item

system_reports/nswAllPearsonY3.csv	
<b>Purpose</b>	<input type="checkbox"/> Output all results of year 3 students, following the fixed-width format prescribed by Pearson for processing.
<b>What to expect</b>	<input type="checkbox"/> A record for each student, covering all responses in all tests the student has sat, with fixed width fields rather than comma-delimited fields.
<b>Look out for</b>	<input type="checkbox"/> Students enrolled in Writing but not Numeracy. <input type="checkbox"/> Scores for individual items add up correctly.

system_reports/nswAllPearsonY5.csv	
<b>Purpose</b>	<input type="checkbox"/> Output all results of year 5 students, following the fixed-width format prescribed by Pearson for processing
<b>What to expect</b>	<input type="checkbox"/> A record for each student, covering all responses in all tests the student has sat, with fixed width fields rather than comma-delimited fields.
<b>Look out for</b>	<input type="checkbox"/> Scores for individual items add up correctly.

system_reports/nswAllPearsonY7.csv	
<b>Purpose</b>	<input type="checkbox"/> Output all results of year 7 students, following the fixed-width format prescribed by Pearson for processing
<b>What to expect</b>	<input type="checkbox"/> A record for each student, covering all responses in all tests the student has sat, with fixed width fields rather than comma-delimited fields.
<b>Look out for</b>	<input type="checkbox"/> Scores for individual items add up correctly.

system_reports/nswAllPearsonY9.csv	
<b>Purpose</b>	<input type="checkbox"/> Output all results of year 9 students, following the fixed-width format prescribed by Pearson for processing
<b>What to expect</b>	<input type="checkbox"/> A record for each student, covering all responses in all tests the student has sat, with fixed width fields rather than comma-delimited fields.
<b>Look out for</b>	<input type="checkbox"/> Scores for individual items add up correctly.

item_printing/itemPrinting.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports one row of response data per student per test item presented within a NAPLAN test <input type="checkbox"/> Detailed information which can be used for ingest into data analysis systems
<b>What to expect</b>	<input type="checkbox"/> Very large report <input type="checkbox"/> This report excludes writing
<b>Look out for</b>	

item_printing/nswItemPrinting.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports item responses for each item responded to, against the student PSI, following NSW TAA format requirements
<b>What to expect</b>	<input type="checkbox"/> A record for each item response
<b>Look out for</b>	

item_printing/qcaa_napo_student_responses.csv	
<b>Purpose</b>	<input type="checkbox"/> Export RRD to QCAA database table
<b>What to expect</b>	<input type="checkbox"/> One row for each individual student item response contained in NAPStudentReponseSet
<b>Look out for</b>	

item_printing/PrintAll.csv	
<b>Purpose</b>	<input type="checkbox"/> Reports all item responses for all tests sat by a student, following NSW TAA and QCEC format requirements
<b>What to expect</b>	<input type="checkbox"/> A record for each student, including all item responses in all tests
<b>Look out for</b>	<input type="checkbox"/> Very large number of columns: the report has a fixed number of columns, and allows 30 items per testlet, with 3 testlets per test (4 for Yr 7 and 9 Numeracy).

NOTE: the items responses are output in the order they occur in the RRD XML. This is not necessarily the same order as the corresponding items occur in the testlet in the codeframe; that needs to be recovered explicitly as the ItemResponse.SequenceNumber. Moreover, items are substituted for other items. As a result, a column in an item response report will not report responses against the same item throughout, and responses need to be collated against items separately if that is required.





## 5.6 XML

**Note: This functionality generates modified RRD XML file/s and it is recommended to be used only after reading the below guidance.**

This function is used to re-extract redacted xml from the Results and Reporting dataset as both a single file per ingest (typically a single RRD) AND a separate XML file per school (based on ACARAID).

The supplied default `naprrql.toml` file by default redacts the following values:

- ☐ Student Item Response
- ☐ Item Correct Answer

for all tests except Writing set.

Run **naprrql.exe -xml** on Windows, **naprrql --xml** on Linux and Macintosh.

**naprrql --xml**: Outputs all ingested NAPRR records in XML format. This outputs a single file for the entire sector, as `out\redacted_xml\sif.xml`.

It is possible to output the XML as one file per school, as well as one file for the entire sector ingested. To do so, enable the report `XMLPerSchoolOutput`, by setting `activated = true` in the file `\config\XMLPerSchoolOutput.toml`. (The per-school set of XML reports is disabled by default.) The content of the generated files is identical to that of the single file for the entire sector. The results will be output in the same format as for the NAPLAN API, and can be used to emulate it:

- ☐ One file per school, `schooldata-{GUID}.xml`, containing `NAPEventStudentLink`, `NAPStudentResponseSet`, `NAPTestScoreSummary`, `StudentPersonal`.
- ☐ One file containing all `SchoolInfo` objects, `schoollist.xml`; this is used to look up the GUIDs for the `schooldata-{GUID}.xml` files.
- ☐ One file containing all `codeframe` objects, `testdata.xml`: `NAPCodeframe`, `NAPTest`, `NAPTestlet`, `NAPTestItem`,

**IMPORTANT:** If you want the XML output to reflect the full contents of the input XML file, you *must* remove all entries from the `XMLFilter` element in the **naprrql.toml** file (the supplied **naprrql.toml** file contains entries to demonstrate the redaction functionality).

The XML follows the SIF/XML schema just like the output of the National Assessment Platform. In particular, elements are redacted by setting their `xsi:nil` attribute to `true`, rather than removing the element from the file. **However it has the following differences:**

- ☐ All multi-object files are wrapped with the single tag `<sif>`, as is done for the NAPLAN API. There are no more specific container elements, such as `StudentPersonals` or `NAPTests` (although the container elements are not used consistently in the source XML anyway). (Such container elements have not been generated by the National Assessment Platform as of 2019.)
- ☐ Empty elements like `<StateProvince/>` will have space inserted between the tag name and the `/>`.

- ❑ If an element in the source file is already marked as redacted or missing, by setting its `xsi:nil` attribute to true (e.g. `<GridLocation xsi:nil="true" />`), it will simply be left out of the XML output.
- ❑ Objects will not necessarily occur in the same sequence in the output file as they do in the source file.

You can set elements to be redacted in the XMLFilter element in the **naprrql.toml** file. The XMLFilter element is an array of paths of elements to be redacted, in XPath notation; e.g.

`/NAPTest/TestContent/DomainBands/Band1Upper,`  
`/NAPTestlet/TestItemList/TestItem[1]/TestItemLocalId.` All paths should be prefixed with `/` followed by the filtered object name.

We have provided some sample values in the **naprrql.toml** provided with the downloaded program.

**IMPORTANT:** If you want the XML output to reflect the full contents of the input XML file, you *must* remove all entries from the XMLFilter element in the **naprrql.toml** file. The supplied **naprrql.toml** file contains entries to demonstrate the redaction functionality, including two sample redaction sets (the second is commented out).

## Section 6 Support

### 6.1 Support

Please contact NSIP directly at [info@nsip.edu.au](mailto:info@nsip.edu.au) or Phone: +61 3 9910 9827 for support

To be notified of updates to the NIAS tools, subscribe to notifications on github:

<https://github.com/nsip/nias2/releases>