Paper SM04

# An Automated Compliance in Project is possible!

Elisa Natale, MSD, Zurich, Switzerland
Thierry Paquet, MSD, Brussels, Belgium

## ABSTRACT

In the pharmaceutical industry, quality and compliance are the two pillars of submissions to authorities. Global and local SOPs design our ways of working to ensure rigor, correctness, and reproducibility. Ensuring compliance (in programming flow, specifications, documentation, testing, etc.) may be challenging for statistical programmers working in big teams and under time pressure.

Program0status tool automates the project compliance checks based on SOPs expectations. It scans the development folder's structure to identify potential discrepancies in files locations, timestamps, content, and naming conventions. A report is created to help the statistical programmer to take corrective actions and make the project fully compliant. This standard tool ensures consistent compliance across projects.

## 1. INTRODUCTION

The development of drugs requires traceability, reproducibility, and efficiency. These are managed by external and internal regulations, which establish boundaries of environment and processes.

The sheer volume of analysis and reporting documents and procedures that need to be followed and manually checked can be time-consuming and prone to errors, leading to missed compliance issues.

The objective of the tool presented in this paper, program0status, is to help the statistical programmers to reach full compliance in a project, by automating the checks and advising specific actions to be taken. By presenting specific case scenarios, details are provided on compliance checks, their interactions and how the system offers advice to easily achieve compliance and reduce risk.

## 2. COMPLIANCE

### 2.1. WHY TO BE COMPLIANT?

To ensure safety and efficacy, many laws and regulations govern the pharmaceutical industry, including those issued by national governments, medical authorities, and drug administrations. To comply with them, pharmaceutical companies develop their own internal regulations and standard operational procedures (SOP), ensuring quality, consistency, and coherence across their projects. Compliance also ensures robustness and trust in the analysis results, allowing the best treatments to be made available to patients. It also helps to streamline the approval process which facilitates the market access. Moreover, by investing in compliance, pharmaceutical companies establish themselves as professional, reliable, and committed to providing top-quality healthcare services.

Non-compliance with official regulations as well as internal procedures and conventions can have severe consequences for pharmaceutical companies. It can result in financial penalties and quality issues that can damage its reputation, brand image and the confidence patients have in the company's products.

### 2.2. WHO IS RESPONSIBLE FOR COMPLIANCE?

While project leaders are responsible for submitting compliant and solid dossiers to authorities, it is the responsibility of each team member to comply with the relevant regulations and SOPs. Mentoring, management, and leadership play a key role in guiding team members.

## 3. HOW TO CHECK COMPLIANCE: PROGRAM0STATUS TOOL

Checking compliance in analysis and reporting of clinical trials can be a lengthy and time-consuming process. Even when working with checklists, ensuring full compliance across hundreds of programs or outputs can be challenging. In such frequent scenarios, oversights or mistakes can occur quickly.

A smart statistical programmer is one who embraces laziness, automating repetitive and extensive checks to establish an intelligent compliance verification process. While some human input may still be necessary, it should be minimized to save time, increase quality, and enable developers to concentrate on more challenging analysis and reporting programming aspects of projects.

The automated compliance *program0status* tool functions as a standalone SAS®-based program. It can be executed as needed, but it is recommended to perform compliance checks throughout the entire development process. Identifying compliance inconsistencies as early as possible is crucial.

The tool compares and cross-checks the defined compliance rules against the project material (programs, outputs, specifications, testing, and documentation). Then, a findings report is generated, providing a clear description of the issues, and suggesting the most appropriate corrective actions to achieve full compliance.

Before proceeding further, it is important to note that compliance is an aspect kept consistent and stable across projects. The rules and principles are standardized and cannot be customized on an ad-hoc basis. Whenever an update is required to address new scenarios, the tool is updated for all projects.

### 3.1. GROUPS OF CHECKS

No matter which company is involved in programming compliance, certain checks are often recurrent across the industry. On the other hand, some checks may be specific to a particular company or department. To take a more generic approach, the checks covered by the tool are presented in categories.

#### 3.1.1. THE RIGHT FILE AT THE RIGHT PLACE

The first requirement for a project to be compliant is having the appropriate material stored in the designated location. This applies not only to tables, listings, figures, or datasets but also includes, among others, program requirement specifications documents, log files, and validation material (programs, inputs, outputs).

To achieve this, the tool needs to gather the following information:

- **The expected files**: The files generated for a project can be either project-specific or standard. To identify them, it is recommended to utilize a tracker sheet to list the non-standard programs to check. For example, if the project involves the creation of ADSL and ADTTE setup programs, they should be listed in the tracker sheet. Once the tracker sheet is complete, the compliance tool has a list of expected project-specific programs to check.

- **The expected locations of the files**: To ensure efficiency, consistency, and traceability, it is advisable to work with a standardized folder structure. The locations of inputs and outputs should remain consistent across projects. To process the expected files, the tool scans the relevant folders, such as program folders, log folders, output folders, and specification folders.

After gathering the needed information, the compliance tool compares the list of files found with the list of expected files. If any expected files are missing, a finding entry is created in the compliance report. Conversely, if any unexpected files are found, the user investigates further to either clean up or update the tracker sheet to include these files.

#### 3.1.2. NAMING CONVENTION

Automation often relies on standardization. One such standardization aspect is naming conventions. The compliance tool can assist and guide users to ensure compliance to the agreed-upon file naming conventions. When comparing expected and found files, there may be cases where two file names are expected to match but do not. In such instances, this discrepancy is reported as a finding. The tool's granularity allows to identify potential issues such as typographical errors in file names, which can explain the non-match. Understanding the root cause of a finding is crucial as it enables users to address issues more effectively.

#### 3.1.3. TIMESTAMPS

Having a compliant list of files at the correct location is not sufficient. Paying attention to the timestamps associated with those files is also important. Consider the following examples of expected timestamp logics:

- The date of the log file should be after the date of the corresponding program.
- The date of validation should be after the date of the program being validated.
- The date of the validation log should be after the date of the program being validated.
- The date of the specifications should be before the date of the validation.

Implementing timestamp compliance helps prevent inconsistencies in validation, specifications, or development versioning. Incorporating the time dimension enhances the guidance from the compliance report.

To ensure the comparability of timestamps, they need to be defined and expressed in a consistent manner. Several prerequisites must be met to enable this check:

- Definition of the data format.
- Definition of the time format: The tool utilizes the 24-hour format.
- Definition of the time zone.
- Other relevant time settings.

Failure to establish these settings uniformly may require additional date/time manipulations before processing, which may not be included in the default code of the tool and may necessitate an update.

#### 3.1.4. HEADER INFO

Compliance also involves examining the information provided in the program headers. They help to identify the program name, its function, the developer identity, the version date, and the history of updates. It plays a crucial role in understanding the program's context, how and when it has been developed. Additionally, it facilitates program use over time and across projects. For this reason, the compliance tool also verifies the different sections of the program's header.

#### 3.1.5. CLEAN LOG

Being compliant includes having clean execution logs. From a designated folder location, the tool can import and scan the log files looking for any error, warning, or note notifications. If a warning is deemed acceptable, such as in the case of statistics divergence issues, the exception should be documented in the tracker documentation. The tool checks for these exceptions before providing any findings related to the log.

#### 3.1.6. TRACKER DOCUMENTATION COMPLIANT

As already mentioned, working with a 'tracker document' is paramount. Regardless of the format, the analysis and reporting process should standardize the tracker structure to facilitate and automate its usage. This document may contain additional metadata about the programs, such as the programmer's name or a brief description of the programs. When standardized, the tool can also verify that all mandatory information has been properly completed.

### 3.2. REPORT

Having a clear and user-friendly compliance report can greatly facilitate and encourage engagement from every team member in compliance efforts. The report should be intuitive, user-friendly, and provide clear messages.

The report is generated in xlsx format and includes two sheets:

- Summary: This tab provides an overview of the number of possible findings identified. The user has the option to edit this tab and document any exceptions that have been agreed upon.
- Details: This tab provides a detailed breakdown of every possible finding.

#### 3.2.1. SUMMARY TAB

This tab aims to provide users with a quick overview of the compliance status.

In Table 1, the report appears as expected when everything is functioning properly. Conversely, if any issues are detected, the report looks like Table 2. The "Comments" column can be utilized to provide explanations for accepted

issues. For example, if the issues pertain to programs that are not yet ready for promotion, they will be addressed at a later stage.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Summary | | Details | | |
| 2 | # programs | Status | # programs | Problem Details | Comments |
| 3 | 26 | Programs ok | . | | |
| 4 | 0 | Programs with some issues | | | |
| 5 | 2 | Programs in archived folders | . | | |
| 6 | | | | | |

Table 1: Example of a “Summary” tab with no findings in the project.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Summary | | Details | | |
| 2 | # programs | Status | # programs | Problem Details | Comments |
| 3 | 26 | Programs ok | . | | |
| 4 | 81 | Programs with some issues | 3 | Problem 01 | |
| 5 | . | | 4 | Problem 02 | |
| 6 | . | | 2 | Problem 03 | |
| 7 | . | | 4 | Problem 06 | |
| 8 | . | | 2 | Problem 07 | |
| 9 | . | | 5 | Problem 08 | |
| 10 | . | | 3 | Problem 09 | |
| 11 | . | | 6 | Problem 10 | |
| 12 | . | | 4 | Problem 11 | |
| 13 | . | | 7 | Problem 12 | |
| 14 | . | | 2 | Problem 21 | |
| 15 | . | | 3 | Problem 22 | |
| 16 | . | | 12 | Problem 24 | |
| 17 | . | | 8 | Problem 98 | |
| 18 | 2 | Programs in archived folders | . | | |
| 19 | | | | | |

Table 2: Example of “Summary” tab with some findings found in the project.

### 3.2.2. DETAILS TAB

For each program included in the project tracker, a clear overview is given in this sheet. An example is demonstrated in Table 3. The first column, labeled 'Suggestions', contains specific details of any identified issue, and suggests the most appropriate action to address them. The remaining columns offer additional program-related details to assist with further debugging actions.

| Summary | Identifiers | | Program | | | | | | Program Request Specification | | | Exceptions | Developer Testing | | | Validation | | | | | | | Approval |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Suggestion | Program Category | Program Name | Program Tracking | Program Availability | Program DateTime | Description | Header Date vs Timesta | Log ok | PRS Availability | PRS Naming Conventio | PRS Date Before Validatio | Exceptions Issue | Developer Tester | DevTest In Tracker | DevTest Availability | DevTest Pgm Date | DevTest Log Date | DevTest Header Date vs Timestan | Latest Version Validate d | DevTest Done After Dvlp | Input Data | Output Data | Program Approved |
| **Problem 02:** Program mentioned in the tracker but not found in any folder. Please create it or remove it from the tracker | | macrotest07 | 1 | 0 | . | Program only in tracker | 0 | 0 | | | 1 | | name1 | 1 | 1 | 01MAR2022:18:00:34 | 01MAR2022:18:20:35 | 0 | 1 | 0 | 1 | 1 | 1 |
| **Problem 02:** Program mentioned in the tracker but not found in any folder. Please create it or remove it from the tracker | | setuptest07 | 1 | 0 | . | Program only in tracker | 0 | 0 | | | 1 | | name1 | 1 | 1 | 01MAR2022:18:12:11 | 01MAR2022:18:20:35 | 0 | 1 | 0 | 1 | 1 | 1 |
| **Problem 02:** Program mentioned in the tracker but not found in any folder. Please create it or remove it from the tracker | | startup-hta-unix-remote | 1 | 0 | . | startup | 0 | N/A | N/A | N/A | N/A | | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **Problem 03:** Program found in validation folder but not mentioned in the tracker and not found in any folder. Please delete validation of the program | | macrotest20 | 0 | 0 | . | | 0 | 0 | | | 1 | | | 0 | 1 | 01MAR2022:18:08:15 | 01MAR2022:18:20:35 | 0 | 1 | 0 | 1 | 1 | |
| **Problem 03:** Program found in validation folder but not mentioned in the tracker and not found in any folder. Please delete validation of the program | | setuptest20 | 0 | 0 | . | | 0 | 0 | | | 1 | | | 0 | 1 | 01MAR2022:18:17:31 | 01MAR2022:18:20:35 | 0 | 1 | 0 | 1 | 1 | |
| **Problem 04:** Program mentioned in the 'exception' tab of the programming tracker but not in the 'Plan and track' tab. If an exception is entered, an entry in the 'Plan and track' tab expected. Please check | | only_in_except_tab | 0 | 0 | . | | 0 | 0 | | | 0 | Missing Exceptions from Plan & Track Sheet: dv12 | | 0 | 0 | | | | | 0 | 0 | 0 | |
| **Problem 05:** Description of the program is missing in the programming tracker. Please add a description | pgmanal | calltest08 | 1 | 1 | 02MAR2022:17:27:13 | | 1 | 1 | See A&R Grid | N/A | N/A | | name1 | 1 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | 1 |
| **Problem 05:** Description of the program is missing in the programming tracker. Please add a description | pgmsetup | setuptest08 | 1 | 1 | 02MAR2022:18:58:18 | | 1 | 1 | 1 | 1 | 1 | | name1 | 1 | 1 | 01MAR2022:18:12:39 | 02MAR2022:19:08:01 | 1 | 1 | 1 | 1 | 1 | 1 |
| **Problem 05:** Description of the program is missing in the programming tracker. Please add a description | pgmsetup | setuptest15 | 1 | 1 | 02MAR2022:19:00:49 | | 1 | 1 | 1 | 1 | 1 | | name1 | 1 | 1 | 01MAR2022:18:15:37 | 02MAR2022:19:08:01 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 3: Example of “Details” tab with some findings found in the project.

## 3.3. KEEPING THE TOOL UP TO DATE

In the event of any procedure or SOP changes, it is crucial to timely update the tool. This proactive measure ensures that new projects adhere to the latest guidelines. Effective communication within different departments ensures that the tool checks are based on the most up-to-date compliance rules.

## 3.4. CASE SCENARIOS

Let's explore real-life scenarios where compliance is not achieved and observe how the tool facilitates the investigation and subsequent resolution of these issues. Let's review screenshots from the "Details" tab of the report, with a specific focus on the columns relevant to each example.

**Case scenario 1: The right file at the right place**

| Suggestion | Program Category | Program Name | Program Tracking | Program Availability | Program DateTime |
|---|---|---|---|---|---|
| **Problem 01:** Program found in pgmanal folder but not mentioned in the tracker. Please add program to the tracker or remove it | pgmanal | call0test1 | 0 | 1 | 28JUN2023:16:36:24 |

In this scenario, the program *calltest1* ('Program Name') was correctly created ('Program Availability' = 1) in the call-programs folder ('Program Category' = pgmanal), on 28JUN2023 ('Program Date Time'), but is not listed in the tracker ('Program Tracking' = 0).

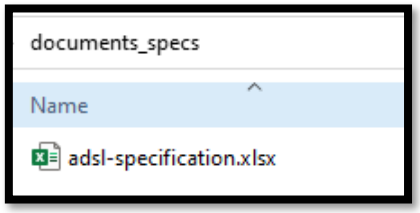
Every program found in the project folders must be mentioned in the tracker, hence there are 2 possible resolutions here:

1. Remove the program from the folder, as it's not needed anymore.
2. Add the program to the tracker and make documentation compliant.

Suggestion' column reflects this message: 'Program found in pgmanal folder but not mentioned in the tracker. Please add program to the tracker or remove it'. If a mandatory document ('PRS' in the example) is missing, the tool gives an appropriate message that can be enriched with additional guidance (such as the naming convention to follow).

**Case scenario 2: Naming convention**

| Suggestion | Program Category | Program Name | PRS Availability | PRS Naming Convention |
|---|---|---|---|---|
| **Problem 12a:** Please check the name of the PRS. It should follow the naming convention setup: <dataset-name>.xlsx | pgmsetup | adsl | 1 | 0 |




In this case, the specification document is found in the right folder. Nevertheless, its name doesn't satisfy the naming convention. The expected name is 'adsl.xlsx'. Hence the message reminds the user how to name such a document.

**Case scenario 3: Timestamps**

| Suggestion | Program Category | Program Name | Program DateTime | DevTest Pgm Date | DevTest Log Date |
|---|---|---|---|---|---|
| **Problem 20:** The validation of this program has run before the tested program date, or the log of the validation is older than the devtest program. Please rerun validation | protocol-macros | macrotest12 | 01MAR2022:18:51:00 | 01MAR2022:18:02:49 | 01MAR2022:18:20:35 |

In this example, the focus is set on the execution sequence. Using the timestamps columns, we can check if the process has been followed in the appropriate order.

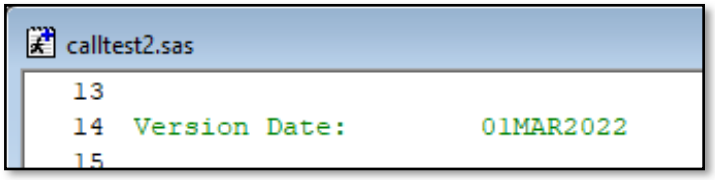
See the following fields:

- Program to be validated: 01MAR2022 18:51
- Validation program: 01MAR2022 18:02
- Validation log: 01MAR2022 18:20

They show that a program finalized on March 1st at 6.51 pm is validated by a code executed before then (on the same day, but at 6.20 pm). There is a potential gap in the process as the program has not been timely validated.
The user needs to check if the validation code covers the latest main program updates and rerun the validation.

**Case scenario 4: Header info**

| Suggestion | Program Category | Program Name | Header Date vs Timestamp |
|---|---|---|---|
| **Problem 06:** The 'version date' in the header does not correspond to the date of the program. Please update the 'version date' in the header in the program | pgmsetup | call0test2 | 0 |


In this example, we can see that the creation date of the file 'calltest2.sas', 02MAR2022, does not match with the date mentioned in the header, 01MAR2022. This will trigger the suggestion "Please update the 'version date' in the header in the program". Such a difference may mean that the program has been modified and saved by mistake and/or the header (history of updates, for example) has not been properly documented. This may lead to misunderstandings or misuse of the program.

**Case scenario 5: Clean Log**

| Suggestion | Program Name | Log ok |
|---|---|---|
| **Problem 10:** The log of the program has some warning, error or note. Please fix them or add a deviation in the programming tracker, column L: 'Exception Number' | calltest12 | 0 |

This situation relates the existence of notifications in a log file with no extra documentation which would have given the useful explanations. A non-clean log may indicate that some issues in the program have not been investigated and documented which possibly produce wrong results.

**Case scenario 6: Tracker documentation compliant**

| Suggestion | Program Category | Program Name | Developer Tester | DevTest In Tracker | Program Approved |
|---|---|---|---|---|---|
| **Problem 14:** The name of the programmer for this program is missing in the programming tracker. Please add a programmer name | protocol-macros | macrotest02 | | 1 | 1 |
| **Problem 15a:** The validation of the program has not been updated in the programming tracker (column K). Please validate the call program and change the status to 'C' **Problem 24:** Please check that the approval status is set to 'approved' 'NA' or blank. If not, please update column 'Customer approval - status' (column V) in the programming tracker | subpop2 | calltest10 | name1 | 0 | 0 |

Finally, some information may not be properly provided to the document tracker, such as missing programmer names or incomplete validation and approver statuses. This can potentially lead to issues when a project is revisited or picked up at a later stage.

## 4. BENEFITS

By utilizing this tool, built upon a standardized structure, process, and conventions, numerous benefits are reaped. First, it enhances stability by identifying and rectifying compliance issues promptly. This reduces the risk of non-compliant deliverables and potential setbacks.

Secondly, the tool contributes to higher quality deliverables. Through its automated checks, it verifies compliance across various aspects of the project material, leaving no room for oversight or error. This results in outputs that meet the required standards and specifications consistently.

Another significant advantage is the consistency achieved between projects. As the compliance tool follows a standardized approach, it guarantees uniformity in implementing rules and principles across different endeavors. This harmonization lends credibility to our work and facilitates future knowledge sharing. Staffs also learn from their lack of compliance. Recurrent findings serve as educational opportunities for teams, helping them avoid non-compliant actions in the future.

The automated nature of the tool brings efficiency to the compliance process. It drastically cuts down the time and effort required for manual checks, allowing programmers and analysts to focus on analysis and programming tasks instead. Efficiency translates to faster project completion and improved productivity.

Additionally, the tool promotes future reproducibility. As new scenarios or updates arise, the compliance tool can be

easily adapted to maintain compliance. This ensures that compliance standards remain up to date and relevant, providing a foundation for consistent and reproducible results.

## 5. CONCLUSION

Compliance is of utmost importance in any project and maintaining a high level of compliance can be a daunting task.

However, with the aid of an automated compliance tool, we can streamline the process and ensure adherence to standards and regulations. We can also greatly improve the compliance process by reducing the time and effort required. These benefits of automation are particularly crucial in larger projects that involve a higher number of programs or documents.

Program0status is designed to identify potential compliance issues, highlight them for review and may even provide with a fix proposal, allowing for faster resolution of issues and reducing the overall risk of non-compliance.

In conclusion, the adoption of an automated compliance tool brings about significant benefits, such as stability, higher quality deliverables, consistency between projects, efficiency, future reproducibility, and time savings. By integrating this tool as part of the process, the desired level of compliance is achieved and maintained while enhancing overall project success.

## CONTACT INFORMATION

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

Elisa Natale
MSD
Email: elisa.natale@msd.com

Thierry Paquet
MSD
Email: thierry.paquet@msd.com