Executable code: what, why and how

Hiren Naygandhi

Roche Products Ltd, Welwyn Garden City, UK

The pharmaceutical industry is continually evolving as it meets the on-going demands of external pressures such as rising development costs, patents, regulatory bodies and patients. Programmers can contribute by, accelerating the time to approval; ensuring patients receive treatment as quickly as possible. One area in particular where programmers can make a difference in accelerating the decision making, is by providing health authorities with executable code as part of the e-submission in the US. At Roche we have developed a standard approach with tools to facilitate this process.

Keywords: Executable Code, Standalone code, FDA, Macros, SAS, E-Submission

Introduction

For a company, the drug development and approval process is worth potentially millions whilst developing a product. However, the bigger picture and vision is always: providing better healthcare earlier for patients. Developing executable code (EC) aims to reduce the amount of time taken in the drug approval process and so achieves the aim of patients getting quicker access to treatments. EC is written to allow it to replicate submitted output independent of:

1. The operating system in which it was developed;
2. The local statistical programming environment in which it was developed.

It is essentially the same code that was developed as part of an analysis; however, it must be standalone code and also be able to run in a standard PC SAS® environment used by the Food and Drug Administration (FDA). All code provided must be completely transparent; there should be no embedded black-boxes or hidden formats.

What is Executable Code?

Within the context of this paper, the points below list our understanding of EC, along with the rationale of why the development of a tool was required:

1. SAS program(s) which were used in the analysis of clinical trial studies that are standalone: they do not depend on macros or formats outside of the program. Within Roche, we keep common formats and macros outside of the programs, so they are called from appropriate external locations. The reason for this is maintainability (if the formats/macros are to be used by more than one program) and efficiency (less code to run within a program). This, though, would cause an issue when submitting to the FDA; these format and code libraries are not available in the FDA. computing environment, so the program(s) must be self-contained.
2. Programs which produce one single output: Again, due to reasons of efficiency and re-usability, we had many cases where one program produced multiple outputs. The reason to keep one output per EC is to allow transparency for an FDA reviewer.
3. Compatible with Windows SAS: Ideally the program should be as platform independent as possible, however, since the FDA use PC SAS, and within Roche, we used a UNIX SAS platform, a need was highlighted to ensure programs were PC SAS compatible, at the very minimum.

EC provides an exact method on how selected outputs are produced from the source data. They provide the FDA with a window on the detailed steps taken to produce those outputs.

Current Guidelines

Current guidelines by the FDA, suggest programs used in analysis do not have to be submitted (unless requested) and only provide formatting details as to how those programs should be submitted.[1] The guidelines contain very little information on which type of programs to submit, i.e. efficacy or safety; primary and/or secondary endpoints; analysis programs and/or TFLs, the number of programs to submit, and so on. If the FDA is stating submission of programs to not be mandatory then why go through the effort to do so?

Why Do It?

There are many reasons to submit EC as part of submissions:

1. E-submission (Esub) deliverables such as define files aid FDA reviewers in identifying programming derivations, in order to verify/replicate analyses, which is where most of their time is taken.[2] Providing EC will make it easier to see exactly how the analysis coding was performed and, in

Roche Products Ltd, 6 Falcon Way, Shire Park, Welwyn Garden City AL7 1TW, UK. Email: hiren.naygandhi@roche.com

DOI 10.1179/1757092112Z.00000000011

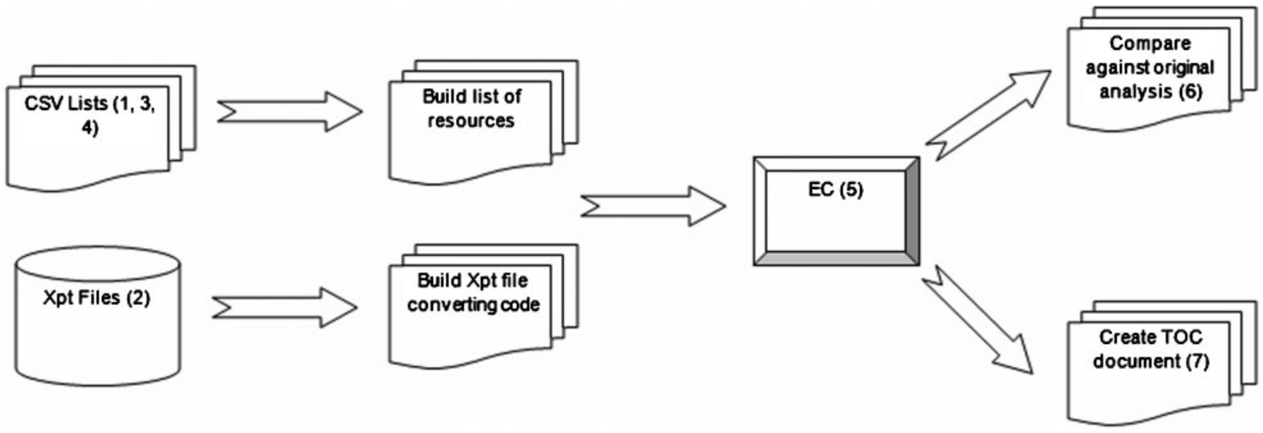

Figure 1 EC creation process. Note: numbers provided in brackets related to the steps identified in the 'Process' section

theory, should facilitate an accelerated decision-making process.

2. In the past, the FDA has come back with requests of providing program code, for example as a filing question. When these types of requests arrive, they need to be worked on and actioned immediately. This would cost the company resource, time and pressure as it is during the critical path. Moving forward, submitting EC as part of submission packages should reduce the risk of program-specific requests.
3. As the industry is moving towards globalized standards and in-house reporting tools, developing EC compliant programs, is something that should be considered.

Once it was decided a tool to create EC was required at Roche, we had to find out what was the quickest and most cost-effective way of developing it.

What We Did

Background

At Roche, it was agreed at a global level that all Esubs, along with the required deliverables, e.g. Define File, Xpt files, etc. will also include key efficacy/safety programs which were used to develop the analysis datasets, as well as the programs used to develop the key efficacy/safety outputs. A specific need arose on a project which was due to be submitted to the FDA, followed by another project that was also due to be filed the following year. As mentioned earlier, the requirements for EC were: the programs must be standalone; produce one output per program; and be Windows SAS compatible. Since the current state of our programs was not fulfilling these criteria and there were multiple submissions that were to include EC, it was decided a tool was required to convert these programs into EC, to be used for these submissions with the long-term vision to enable global use of the tool.

Methodology

A couple of SAS options were considered when deciding how best to convert the current programs that are available into EC. The first was to use the SAS `MPRINT` option and generate a program from the contents of the log. `MPRINT` displays all SAS code which was used to run a program (including macros that are saved internally or externally). This feature allows a user to generate SAS code from the Log. This methodology was dropped as it was considered too complex and time-consuming.

The alternative was to use another SAS option, `RTRACE`. `RTRACE` can be used to generate a list of the resources that are read or loaded within a SAS session.[3] This feature allows a user to create electronic readable files containing the list of resources used by a program, i.e. datasets, macros, files. Using this list, all these dependencies can be constructed into one EC, per program. This was the methodology that was decided on as the most time-saving option.

A full list of resources enables the identification of all:

1. macros called by the program;
2. SAS datasets used by the program;
3. external files (.TXT,.CSV etc.) used by the program (these could be Output (results) files, Data source files for value-added datasets, etc.).

Once the program(s) have been identified, all that is required is to:

1. concatenate into one single program all the macro programs called (without their include statements);
2. provide a mechanism of allocating the SAS data resources and external data resources used;
3. append the program that needs to be run.

When the resulting EC program runs, all the relevant data is allocated, all the required macros are compiled and then the program can run successfully. It is worth noting, there may be a critical path associated with the order of the macros as they are compiled but this can be determined from the order they appear in the SAS `RTRACE` log. The system employed at Roche automates these steps.

Process

The process overview is described below (along with the steps identified in Fig. 1):

1. Identify programs to be submitted: this is one of the most important tasks (and in hindsight quite cumbersome — more on this later) where a decision

must be made on which program(s) need to be submitted. The decision makers in this stage are the regulatory and biostats representatives; however, statistical programmers are key in recommending the approach to take. The most important point to note here is any programs that are chosen should be considered key to the submission/filing i.e. the most common case being the programs used to display the primary and/or secondary endpoints for the study/studies. A suggestion was made of 'why not send everything?', however, in light of optimizing process value and saving time, a smart risk approach was taken to send only EC that would aid the submission. These would be inputted in the EC tool as a CSV file.

2. Identify Xport files: data sent in submissions, are always sent as Xport (xpt) files. Though the FDA would typically have software to convert these xpt files into SAS datasets, an additional step was taken to do this conversion mechanically within the process, based on the xpt files required for a program. This would allow the FDA Reviewer the choice to use datasets they already may have converted or use the xpt files that have been sent as part of the submission.
3. Identify formats program: generating the format code is achievable by using the SAS `PROC FORMAT` using the `CNTLOUT` option; however, `RTRACE` is unable to provide a list of formats used. This does provides a bottleneck in code development as it has to be done manually (see Challenges section). As a result, if the program requires a format that was developed in another program, this would also need to be provided as an input (as the `RTRACE` option would not be able construct the format code from the format catalog) and so inserted into the EC after development. Note: For any large formats such as medical dictionary values, the values are decoded on the submitted XPORT data sets.
4. Provide an appropriate SAS GLOBAL statement (and provide appropriate variable assignment): To ensure the macros used can refer to values pre-defined in the local environment.
5. Build EC using list of resources: Using the programs identified as an input, a list of the resources are produced. Using this list allows a user to build the EC. The EC developed would consist of (in the order they would appear in the EC):
   - a standard header stating the name of the program; describing the use of it, what datasets are required to run it, what macros are required, etc.;
   - a set of macro variable `LIBNAME` options available to the Reviewer, where they set them to a particular value: to choose whether they wish to use xpt files or SAS datasets, location of where they have stored these data and the location of where they wish to store the output produced;
   - standard code to copy over datasets/convert xpt files into datasets;
   - constructed formats code (if provided as an input);
   - code for the compilation of macros required by the EC;
   - original program code;
   - standard code to save the output produced
6. Test and check EC in PC SAS: Once these components are combined in the program in the correct order the remaining step is to ensure the code is syntactically correct for the FDA environment operating system i.e. testing it in PC SAS for any errors or warnings and ensuring it matches the output that was produced in the original analysis (the Logs and Compare programs would be saved for audit purposes). In Roche, the testing step is straightforward as notwithstanding the file allocation syntax in `FILNAME` and `LIBNAME` statements (all that has to be changed is the allocation statements at the top of the code for the program to run in both environments), the UNIX environment and the PC-SAS environment have a large degree of communality. This step would be more difficult moving from, for example, an IBM (EBCEDIC) environment as there is less commonality however, the lack of commonality can be avoided by using SAS Connect software or by using common code where the operating systems differ e.g. using `LE` rather than <= for less than or equal to, using `!!` instead of `||` for concatenation, using `NE` instead of ¬= for is not equal to etc. Note: The EC tool itself does not need to be validated, as once the EC output and original analysis output has been compared and match, the tool becomes self-validated.
7. Develop a table of contents (TOC): Once all EC has been developed, tested and is ready, a TOC document would be created and would describe steps the FDA Reviewer would need to take in order to run the code, along with information on each EC provided, including name, purpose, datasets/xpt files used and so on.

See Fig. 1 for the process overview.

Challenges

During the development of the tool, there were many challenges. The major challenges are listed below.

Identifying programs required for submission

One of the initial challenges faced was identifying which programs should be used to send as EC, as part of the submission package. The guidance available was to choose key outputs in supporting the efficacy/safety objectives of the study/studies, i.e. the primary/ secondary endpoints, and work backwards to identify the programs which generated these outputs. However since this was a fairly new process at the time it made it difficult to find guidance on what has been done historically. Also, the decision made was a subjective one between the Regulatory and Biostats functions, since it was based on their judgment and knowledge of the study/studies. The recommendation was to meet up with both functions early and discuss which to include. The benefit would be the decision can be made early and in agreement with all parties concerned.

Esub requiring more datasets

During the development of EC, we realized although the base data was provided, not all dependencies were being provided as part of the datasets being sent to the FDA, e.g. interim datasets with lab value conversions; external text files and so on, as part of the eSub. This does not imply what was being sent in the eSub was incomplete, since the requirement is met by sending source and analysis xpt files. However, in order to run the EC, we would need to send all the data/files which are required. Though this required more work to be done on the eSub, it also meant the EC tool was a good cross-checking device to ensure all datasets/files have been included as part of the eSub package, which were required by the program.

Incomplete resources

The use of `NOSOURCE` within code, caused issues, as whenever this was used, the `RTRACE` feature would not be able to get all the resources which were required by the program from that point onwards, where `NOSOURCE` was appearing. This would mean the EC would be produced but may not contain all the resources that are required, i.e. would be incomplete. The solution was to remove the code where `NOSOURCE` was used. Programmers use this option to not clutter the SAS Log (as it suppresses anything to go to the Log once called), though after the finding, it was recommended within the Programming team at Roche to avoid the use of it as part of the Coding Standards.

Too many resources

The use of dictionary tables also caused issues. We came across a scenario where `SASHELP.VCOLUMN` was used in a macro as part of a data statement. The issue was the EC tool read all the resources that were part of this special dataset in the SAS session at the time. For example, where a program, required 10 datasets to run, the EC produced would state 900 datasets are required! The solution was to convert the use of `SASHELP.VCOLUMN` to `DICTIONARY.TABLES` within a PROC SQL step rather than a Data Statement, which is a more efficient way of using Dictionary tables.[4] The suggestion is to implement its future use as part of the Coding Standards.

Using double-programmed reports as opposed to firstline

The requirement is to produce EC that would be the same as what was produced in the original analysis. Since there are no other rules to define which program that may be, we took a novel approach in one of the projects, to use QC (or double-programmed) programs. The problems we were finding with using firstline programs were:

1. Since the firstline program was using standardized macros, it took a long time to run.
2. The EC became very long in code as it was using a lot of utility macros that would make it difficult to search for relevant code in the EC program or the SAS Log.

Since the QC programs produce the same numbers/output as the firstline, it would save much time to use them. A possible drawback of using QC programs for EC is even though they follow SAS coding standards the output produced may just appear as a data dump, since the emphasis may be to match what is produced in the firstline output and not necessarily focus on the display/layout of the output (though this would depend on the project rules/guidelines). This, however, is a fairly trivial issue: if the output displays the correct content, it has produced what was required. It also means, if there is any electronic reconciliation of datasets as part of a QC strategy (e.g. `PROC COMPARE`) then it must be removed from the final EC. This may seem as an undesirable step to remove code, however, the final dataset/output will be what is required. Anything after that point is not needed.

Other issues that occurred were considered minor, e.g. use of commenting not always clear or complete; runtimes on large (e.g. over 12 million records) datasets could take a long time to run. This in turn was used to identify poor programming practice and provided indirect benefits of improving future programming techniques.

Metrics

The EC tool took 4 weeks to build, including a submission of six programs. Further time was planned for its development; however, using the EC tool for a submission accelerated the development immensely and also identified issues such as missing formats and using data steps with the `SASHELP` tables. This was safe to do so as the output could always be validated against the original reported output and was still quicker than writing a PC SAS equivalent program manually.

The initial recommendation to produce EC based on using this tool was 2 programs per day (this would include full testing of the ECs). Since this was a tool we had to develop from conception, there was no benchmark to measure the time taken to develop EC for a program. As further submissions were taking place, more teams were using the tool and with the greater experience gained; users were able to develop 10–15 within one day, subject to run times and project specific issues encountered.

The EC tool developed has been used in six submissions to the FDA. In terms of feedback based from the FDA, none have been received to date. Though that means we still do not know what the requirements are for EC, it also means no questions have come back with regards to what was submitted, ergo, no news is good news.

Future

If more time was available to work on the tool, an improvement we would have tried to make would have been to add a Graphical User Interface (GUI) to the front end of the tool. This would have made the task much easier to follow and complete. Another area we would have looked into further was formats. The aim would have been to find a more mechanized (and less manual) approach to adding them to the program.

Moving forward, the company has developed (and continues to) a global tool that contains a standard suite of reporting macros. The tool will also contain a functionality which will enable EC to be produced concurrently; hence, it will produce EC fairly quickly and replace the current EC tool (hence there are no plans to make future improvements on the EC tool), however, in the meantime, the current EC tool will be used. Note: The approach taken globally is to use a combination of `MFILE` and `MPRINT` which goes against the methodology this Paper discusses, however, the decision taken to use `RTRACE` prior to development of the EC tool was justified at the time — as it would have required a lot of re-work on existing programs.

Conclusion

We work in an industry which is very heavily regulated, due to the nature of Clinical Trials. It is well documented that it takes a very long time for a drug to reach the market from discovery, on average 2–12 years and costs \$1.8 billion to take a new compound to market.[5] Taking the time and cost into consideration, patients who are waiting for these drugs suffer the most as treatments they require could be available but not on the market as the FDA must approve the use of them. This is where, Statistical Programmers in the industry, can make a difference.

A tool was developed at Roche that enabled EC to be generated and provided in a standard and acceptable way to the FDA. The development of the tool had many challenges, though once overcome; it proved to be a time-saver on producing EC compliant programs manually as well as being available as an option in the event of an FDA requested program code.

To date, there have been no follow-ups or requests, on submissions that have been made with EC. Based on the opinions of the Author, this would suggest the FDA is content with the EC provided and so it has proved a worthwhile practice. While there has been no direct confirmation on the submission of EC, the fact that Roche is implementing a global tool which will have this feature, means we are better prepared for providing these as part of submissions as well as any FDA requests that come in.

Taking into consideration what we have learnt about EC, the next steps would be to try to find out if EC is something that aids FDA Reviewers in their job. Even if this is not a requirement, if there is evidence to suggest that it is desirable then it would mean we are moving in the right direction.

Acknowledgements

Special thanks to Jim Elder, James Gallagher and Colin Brown for their review and insights of the topics covered.

References

1 Center for Drug Evaluation and Research (CDER), Center for Biologics Evaluation and Research (CBER), Center for Devices and Radiological Health (CDRH), Center for Food Safety and Applied Nutrition (CFSAN) & Center for Veterinary Medicine (CVM), 2003. Providing Regulatory Submissions in Electronic Format – General Considerations. *FDA*, [online]. Available from: http://www.fda.gov/RegulatoryInformation/Guidances/ucm124737.htm [Accessed 6 February 2011].

2 Vachal R. FDA Reviewers as Ultimate End Users: Using the SAS System to Construct e-Submissions that Actually Facilitate the NDA/BLA Review Process, PharmaSug, 2001.

3 SAS Support, 2012. RTRACE System Option: Windows. SAS [online]. Available from: http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#win-sysop-rtrace.htm [Accessed 09 September 2012].

4 SAS Support, 2012. Accessing SAS System Information by using DICTIONARY Tables. SAS [online]. Available from: http://support.sas.com/documentation/cdl/en/sqlproc/62086/HTML/default/viewer.htm#a001385596.htm [Accessed 3 September 2012].

5 ABPI, 2012. Code of practice for the pharmaceutical industry. Prescription Medicines Code of Practice Authority (PMCPA) [online]. Available from: http://www.abpi.org.uk/our-work/library/guidelines/Documents/ABPI_Code_2012.pdf [Accessed 1 July 2012].

Recommended Reading

1. Widel M, Zhou J. Techniques for creating reviewer-friendly SAS programs, PharmaSug, 2006.
2. Todd M. Using SAS Enterprise Guide to Create Standalone Programs, North East SAS Users Group, 2010.