PharmaSUG 2016 – Paper IB04

Programming checks: Reviewing the overall quality of the deliverables without parallel programming

Shailendra Phadke, Baxalta US Inc., Cambridge MA
Veronika Csom, Baxalta US Inc., Cambridge MA

Abstract

The pharmaceutical and Biotech industry trend is slowly shifting from programming deliverables in house to outsourcing programming and data management responsibilities to Clinical Research Organizations (CROs) and Functional Service Providers (FSPs) using their knowledge and expertise. While the responsibilities are being handled by these external sources the programming lead at the pharmaceutical company is still accountable for the quality, accuracy and CDISC compliance of created deliverables. It's a daunting task to check and confirm the accuracy without actually performing parallel programming when presented with a tight timeline, certain key programming checks will assist in checking the overall quality of the deliverable. This paper illustrates a list of such programming and CDISC compliance checks that will aid a reviewer in assessing a deliverable efficiently. After utilizing these checks, the reviewer can ascertain the quality and compliance of a deliverable with greater confidence and easily pinpoint the issues and errors. While these checks will aid in reviewing large portion of deliverables, more specific checks may be needed due to the complexity of a given study.

Introduction

When a data transfer (which might consist of raw data, SDTMs, ADaMs and/or TFLs) is received for review from an external service provider, it can first be overwhelming, especially when the scope of the study is large and complex. It is an even bigger challenge if it is a single reviewer working with a tight timeline.

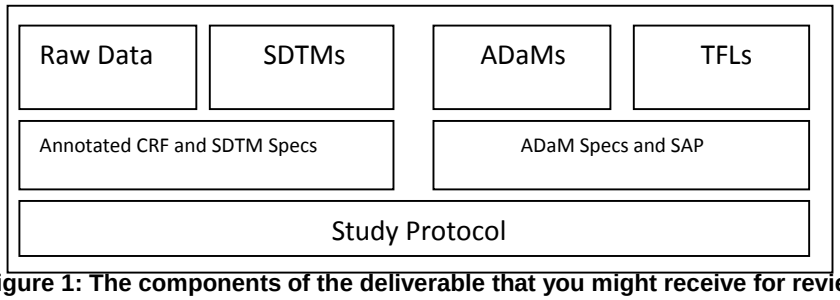


Figure 1: The components of the deliverable that you might receive for review along with the supporting documents.

Prerequisites before starting the review:

As a reviewer you are expected to know the study design, the primary and secondary endpoints and the study schedule, hence reading the study protocol and Statistical Analysis Plan are a requisite. Creating a tracking sheet before starting the deliverable review can be very useful for:

a. Itemizing and keeping track of all the issues.
b. Communicating these issues to the external service provider.
c. Tracking whether or not the issues are being resolved.
d. Monitoring any outstanding issues.

Generally, when a programmer is given the task of programming either SDTMs or ADaMs, he makes himself familiar with the specification documents provided along with supporting documents (i.e.

annotated CRF or the SAP) before starting. When reviewing deliverables it is better to review components of the deliverable and supporting documents in parallel and cross check the numbers.

Review Process:

STEP 1: Reviewing the annotated CRF.

Check if all the fields are annotated on the CRF. These fields should either be correctly mapped to the respective SDTM domains or are clearly annotated as “NOT SUBMITTED”.

STEP 2: Reviewing the SDTMs.

a. Check if all raw data have been mapped. For most of the domains there is a 1:1 mapping between the raw data and the SDTMs. This check can be as simple as comparing the number of records in the raw data and SDTMs.
b. Use the Open CDISC validator report to check if the SDTMs are in compliance with the CDISC standards.
c. Check for outliers, especially in the lab data using proc univariate.
d. If Findings About or Related Records special purpose domains are used; check if the linking is implemented correctly.

The above two steps explain the basic review that should be done for the SDTM.

STEP 3: Reviewing the ADaMs.

Analysis datasets, as the name suggests should be created only if they will contain derived information needed for the creation of the TFLs which is not readily available in an SDTM domain. So as an example if a listing or a summary table is required by the SAP which can be programmed using the SDTM, then ADaM need not be created.

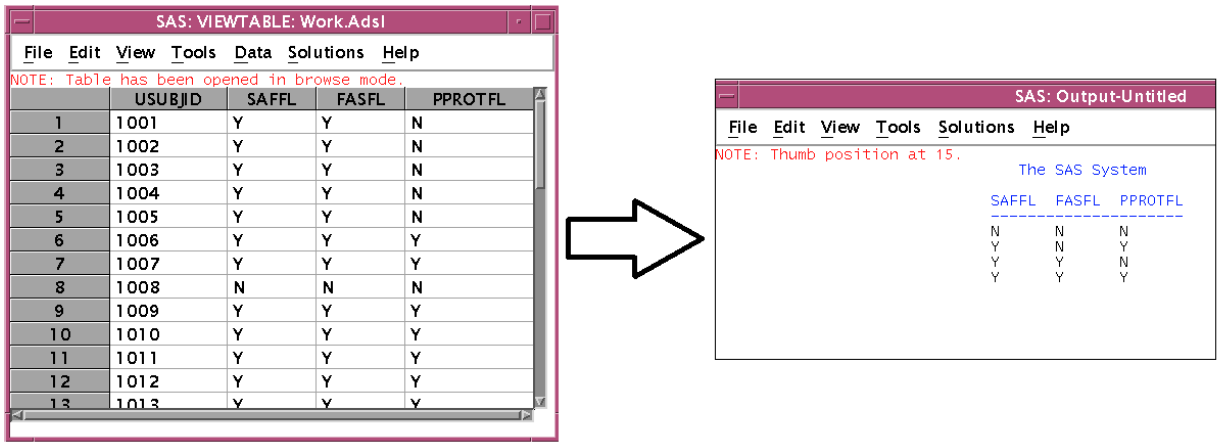
As a starting point, you should always begin with reviewing the ADSL (Subject-Level Analysis Dataset) because the ADSL is always a component in creating all other ADaMs. The resulting analysis datasets should not be considered as individual entities. These datasets are derived bearing in mind the study endpoints and the safety analyses. When reviewing these datasets, the associated TFLs should be reviewed in parallel. The ADaMs and TFLs review can be grouped by safety, primary endpoint and secondary endpoints.

Depending on the study, the therapeutic area and the phase; the efficacy datasets may be different, however most studies will have ADSL, ADAE (Adverse Event Analysis Dataset) and analysis datasets following BDS (Basic Data Structure). We will concentrate on the key points which will be helpful in reviewing these datasets.

Subject-Level Analysis Dataset Review:

A good starting point is checking the number of subjects in different analysis populations. Issues such as contradicting population flags can be detected easily by creating a distinct value list of all the population flags in ADSL. This can be implemented by the following code.

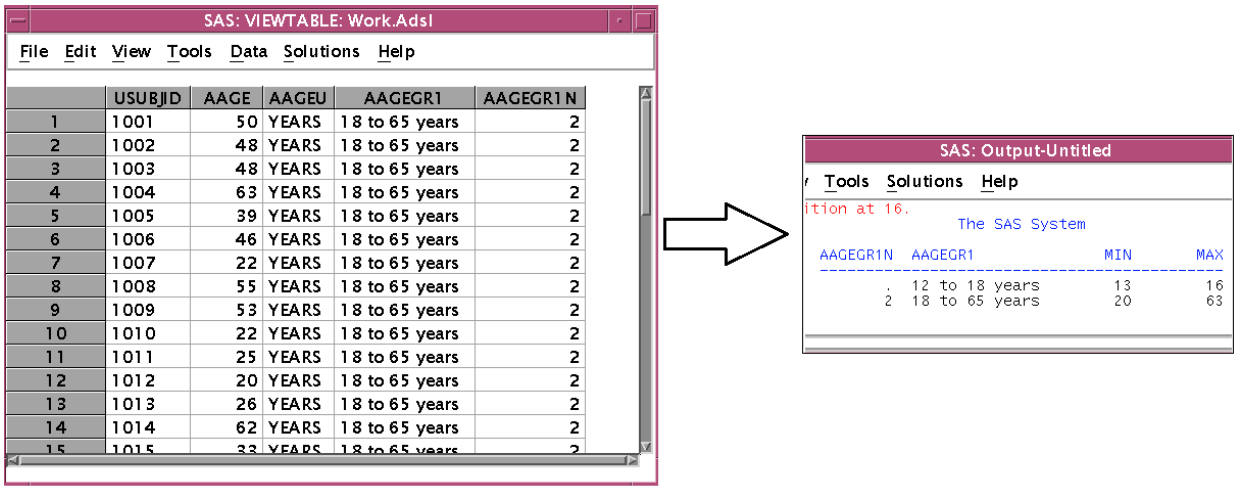
```
proc sql;
select distinct SAFFL, FASFL,PPROTFL from adsl;
quit;
```



Display 1: Input dataset and the output of the above code.

Reviewers should check the accuracy of the group assignment and numeric mapping of the grouping variables. The example code below will check if the age group assignment was implemented appropriately.

```
proc sql;
select AAGEGR1N,AAGEGR1,min(AAGE) as MIN, max(AAGE) as MAX
from adsl
group by AAGEGR1N,AAGEGR1
;
quit;
```



Display 2: Input dataset and the output of the above code.

Further checks that should to be done are assignment of trial treatment to ARM along with related variables such as planned and actual treatment in each epoch, treatment duration and study periods. Additional checks for the treatment and study start and stop dates should be performed, especially for continuation studies.

Adverse Event Analysis Dataset Review:

In ADAE review the main focus is on identifiying missing values for variables used in the analysis (e.g., severity, causality, seriousness, etc.). If there are any missing values; the reviewer will need to confirm if they are handled in accordance with the SAP. A simple distinct value list can be an easy and useful way to start. Below a sample code with output.

```
proc sql;
select distinct AEREL,AREL,ARELN,RELGR1,RELGR1N from adae order by RELGR1N;
quit;
```

SAS: VIEWTABLE: Work.Adae

File Edit View Tools Data Solutions Help

	USUBJID	AEREL	AREL	ARELN	RELGR1	RELGR1N
1	1001	PROBABLY RELATED	Probably Related	3	Related	1
2	1001	POSSIBLY REALTED	Possibly Related	2	Related	1
3	1001	UNLIKELY RELATED	Unlikely Related	.	Not Related	0
4	1002	UNLIKELY RELATED	Unlikely Related	.	Not Related	0
5	1002	PROBABLY RELATED	Probably Related	3	Related	1
6	1004	NOT RELATED	Not Related	0	Not Related	0
7	1004	UNLIKELY RELATED	Unlikely Related	.	Not Related	0
8	1008	UNLIKELY RELATED	Unlikely Related	.	Not Related	0
9	1009	PROBABLY RELATED	Probably Related	3	Related	1
10	1010	NOT RELATED	Not Related	0	Not Related	0
11	1010	POSSIBLY REALTED	Possibly Related	2	Related	1

SAS: Output-Untitled

File Edit View Tools Solutions Help

The SAS System

AEREL	AREL	ARELN	RELGR1	RELGR1N
		.		.
UNLIKELY RELATED	Unlikely Relat	.	Not Related	0
NOT RELATED	Not Related	0	Not Related	0
POSSIBLY REALTED	Possibly Related	2	Related	1
PROBABLY RELATED	Probably Related	3	Related	1

Display 3: Input dataset and the output of the above code.

Another important check in ADAE is the implementation of the link between the AE and associated domains like EX, CM and DS. In cases where there are only a few records affected by the link then a visual check can be sufficient, otherwise a programming check is needed. Depending on the number of occurrence, this strategy can also be utilized when handling partially unknown AE start and end dates.

Serious Adverse Event specific attributes such as hospitalization or death related data that are missing should be further investigated as these are a key part of the safety review. Utilizing a simple code it is easy to determine if there are SAE records with any missing attributes. Another simple check can be to verify if SAE specific variables are only completed for SAEs and not for non-serious events.

To verify if all the treatment emergent AEs are flagged appropriately, the AEs which have not been flagged should be checked manually by comparing the dates of AEs with the corresponding treatment start dates.

Basic Data Structure ADaM Review:

The first part of the review of datasets following BDS structure is to verify if ADaMIG defined variables are used correctly. From a structural aspect it is important to check the proper usage of PARAMTYP, DTYPE, BASETYPE, AVALCATy, ANLzzFL and CRITyFL variables. This will ensure that the analysis dataset is following a vertical structure.

The implementation of PARAMTYP, DTYPE and BASETYPE variables can be confirmed by reviewing the specifications. Once you are sure that the variables are defined correctly, you should check the implementation in the dataset. For example if the possible DTYPE values are: LOCF, WOCF, AVERAGE then you should check if the records have been derived appropriately.

To review the implementation and derivation of AVALCATy and CRITyFL, the following code can come in handy. This code will provide an easy way to check the assignment between analysis values and analysis categories unless there is a long list of possible combinations.

```
proc sql;
select distinct AVALCAT1,AVAL,AVALC from bds order by AVALCAT1, AVAL;
quit;
```

SAS: VIEWTABLE: Work.Bds

File Edit View Tools Data Solutions Help

NOTE: Table has been opened in browse mode.

	USUBJID	PARAM	AVAL	AVALC	AVALCAT1
1	1001	Number of Infusion to treat bleed	4	4	4
2	1002	Number of Infusion to treat bleed	1	1	1
3	1003	Number of Infusion to treat bleed	1	1	1
4	1004	Number of Infusion to treat bleed	3	3	3
5	1005	Number of Infusion to treat bleed	6	6	>=4
6	1006	Number of Infusion to treat bleed	4	4	4
7	1007	Number of Infusion to treat bleed	8	8	>=4
8	1008	Number of Infusion to treat bleed	6	6	>=4

SAS: Output-Untitled

File Edit View Tools Solutions Help

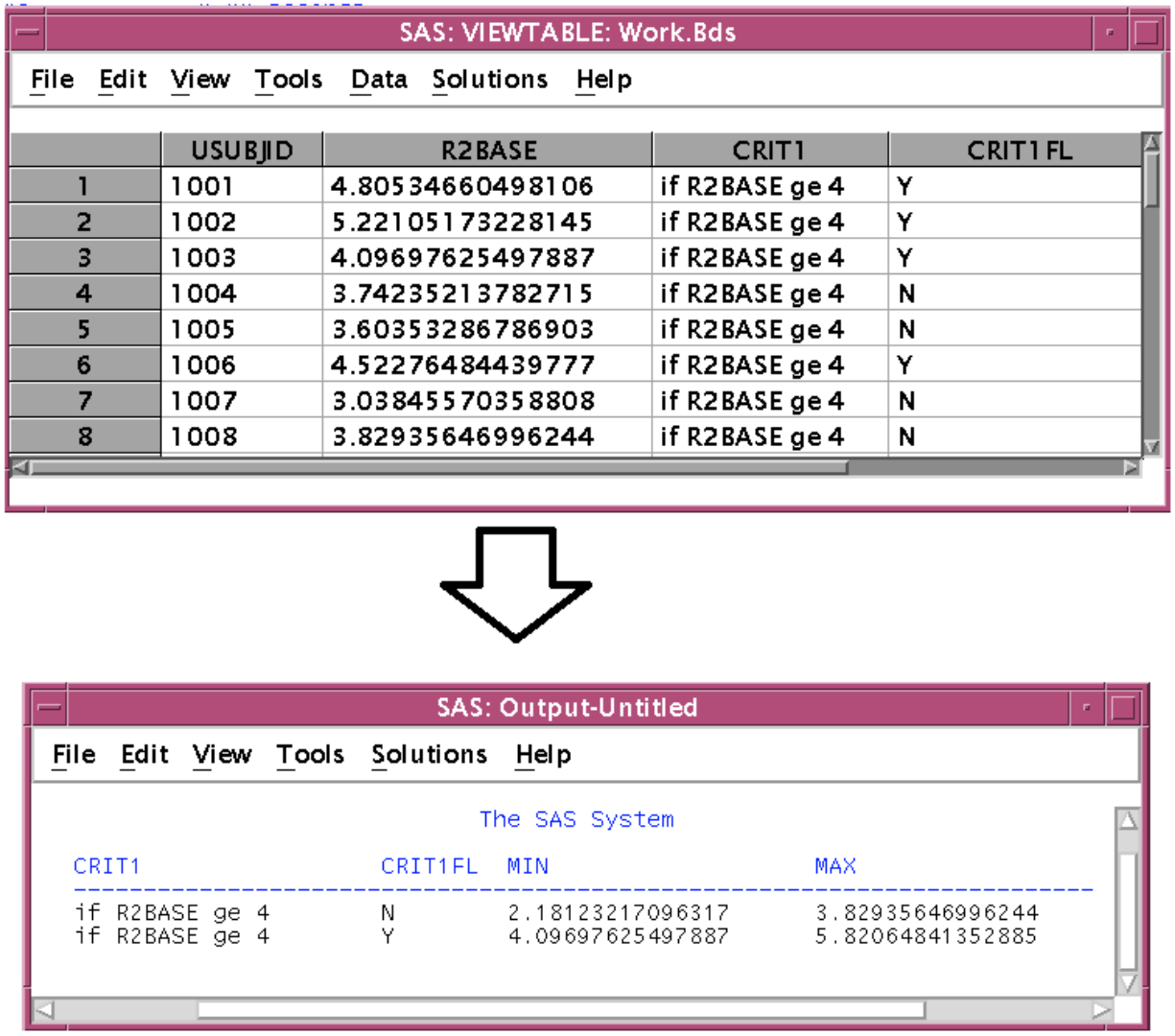
The SAS System

AVAL	AVALC	AVALCAT1
1	1	1
2	2	2
3	3	3
4	4	4
5	5	>=4
6	6	>=4
7	7	>=4
8	8	>=4

Display 4: Input dataset and the output of the above code.

CRITyFL defines either criteria around some threshold values or study specific conditions; hence a code similar to the one utilized to confirm the age group assignment can be useful and supportive in oversight. Below is a sample code created to review a criterion flag that is indicating if a titer increased at least 4-times from baseline. The code will assist in confirming the correct assignment of defined criteria.

```
proc sql;
select CRIT1,CRIT1FL,min(R2BASE) as MIN,max(R2BASE) as MAX
from bds
group by CRIT1,CRIT1FL;
quit;
```

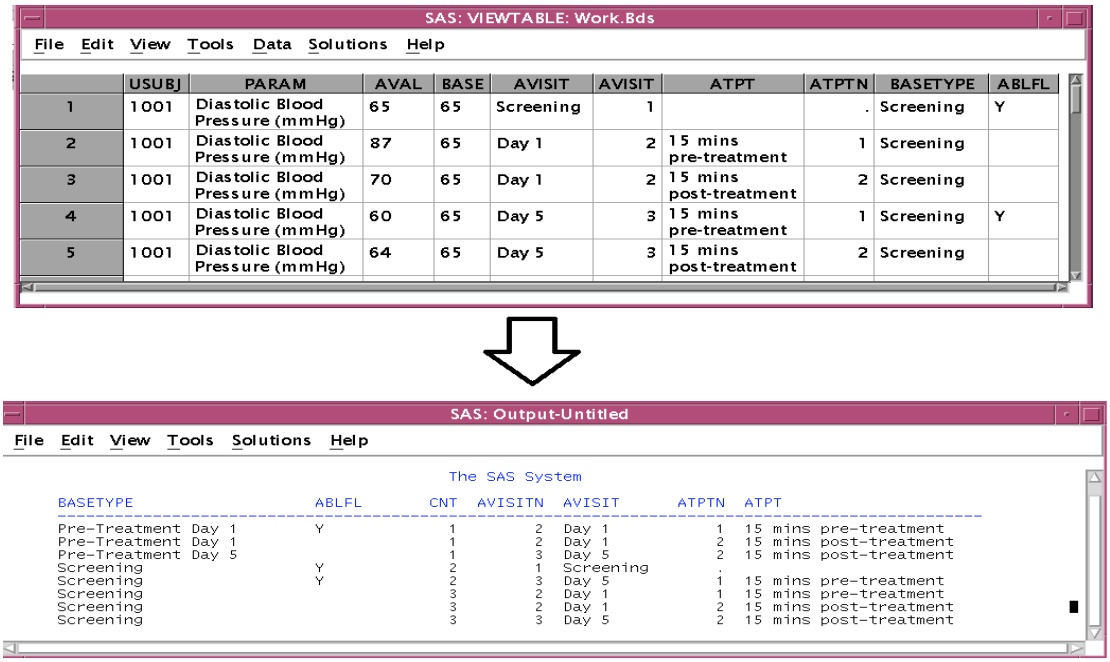


Display 5: Input dataset and the output of the above code.

Baseline flag and related variables (e.g., CHG, R2BASE, SHIFT, BASETYPE, etc.) should be kept in focus, especially if a result is analyzed in comparison to more than one reference value. In such cases not only should the derivation and the content of the variables but also the accuracy of the dataset

structure needs to be evaluated against the CDISC ADaMIG principles. The following sample code can help in this review.

```
proc sql;
select distinct BASETYPE,ABLFL,count(distinct AVISIT||ATPT) as
CNT,AVISITN,AVISIT,ATPTN,ATPT from bds
group by BASETYPE,ABLFL
order by BASETYPE,ABLFL desc,AVISITN,ATPTN;
quit;
```



	USUBJ	PARAM	AVAL	BASE	AVISIT	AVISIT	ATPT	ATPTN	BASETYPE	ABLFL
1	1001	Diastolic Blood Pressure (mmHg)	65	65	Screening	1		.	Screening	Y
2	1001	Diastolic Blood Pressure (mmHg)	87	65	Day 1	2	15 mins pre-treatment	1	Screening	
3	1001	Diastolic Blood Pressure (mmHg)	70	65	Day 1	2	15 mins post-treatment	2	Screening	
4	1001	Diastolic Blood Pressure (mmHg)	60	65	Day 5	3	15 mins pre-treatment	1	Screening	Y
5	1001	Diastolic Blood Pressure (mmHg)	64	65	Day 5	3	15 mins post-treatment	2	Screening	

BASETYPE	ABLFL	CNT	AVISITN	AVISIT	ATPTN	ATPT
Pre-Treatment Day 1	Y	1	2	Day 1	1	15 mins pre-treatment
Pre-Treatment Day 1		1	2	Day 1	2	15 mins post-treatment
Pre-Treatment Day 5		1	3	Day 5	2	15 mins post-treatment
Screening	Y	2	1	Screening	.	
Screening	Y	2	3	Day 5	1	15 mins pre-treatment
Screening		3	2	Day 1	1	15 mins pre-treatment
Screening		3	2	Day 1	2	15 mins post-treatment
Screening		3	3	Day 5	2	15 mins post-treatment

Display 6: Input dataset and the output of the above code.

Lastly to avoid surprises later on; it is very useful to check the distribution of primary and secondary endpoint parameters. As an example if all conversions to standard units were implemented correctly and that there are no implausible results or outliers present can be checked. Checking the distribution of derived analysis endpoint values can also be useful in performing this review; for this purpose a simple univariate procedure can be utilized.
If applicable, handling of all missing values, or unscheduled visits should be checked so that such data are not ignored in the analysis.

CONCLUSION:

As the focus of the Biotech companies is shifting towards outsourcing the actual programming work and building partnerships with CROs, the programmers of the Biotech companies need to change their approach towards studies. Due to programmer being responsible for more studies and having to work with tight timelines; the programmers need to be more innovative towards their everyday work.

1. This paper gives a brief overview of the scope of work and complexity that can be involved, when it comes to the review and oversight of a deliverable.
2. The checks mentioned in the paper will assist in saving time when performing general review tasks, so more time can be spent on reviewing the more complex derivations and endpoints.

3. The best practices outlined in the paper support a more effective, efficient and complete oversight.
4. The code snippets can be used as a skeleton code to build a list of more study specific programming checks.
5. This paper can be used as a starting point or a guideline to compile a more comprehensive and project specific checklist for the review.

References

CDISC SDTMIG v3.2
http://www.cdisc.org/sdtm

CDISC ADaMIG v1.0
http://www.cdisc.org/system/files/all/standard/application/pdf/adam_implementation_guide_v1.0.pdf

CDISC ADaM for ADAE
http://www.cdisc.org/system/files/all/standard_category/application/pdf/adam_ae_final_v1.pdf

Acknowledgement

The authors would like to thank Manuela Koska and Elizabeth Staron from Baxalta for their time and valuable feedback.

Contact Information

Shailendra Phadke, Senior Programmer Analyst,
Baxalta US Inc., Cambridge MA 02142.
Shailendra.Phadke@baxalta.com

Veronika Csom, Senior Programmer Analyst,
Baxalta US Inc., Cambridge MA 02142.
Veronika.Csom@baxalta.com