PharmaSUG China 2021 - Paper AD-020

Title/footnote Auto Extraction from TLG shell to Program Tracker

Jiapeng He, BeiGene, Beijing
Tingting Zeng, BeiGene, Beijing

ABSTRACT

Manually copy metadata info (e.g., title, footnote, source data, analysis set) from Table, Listing, Graph (TLG) mock shell to programming tracker is always time-consuming and inefficient in programming work. However, an VBA Object called “Selection” in Microsoft (MS) Word can be applied to entirely scan and identify different component in a standardized TLG shell whose file extension is DOC, DOCX or RTF.

This paper will introduce that, by raising a few rules to stabilize the format of the TLG shell, we can automatically identify and copy TLG title, population set, footnotes, source data from the TLG shell to a programming tracker, even if an TLG output is referred to other output in the TLG shell. Moreover, to help user easily standardize the TLG shell, VBA add-in object can be applied in MS Word to help format/valid selection text in TLG shell. And a method of quickly merging the updated info in TLG shell to the existing program tracker will also be introduced in this paper.

INTRODUCTION

VBA is a popular computer language that is widely used in pharmaceutical industry for automation within MS Excel, Word, PowerPoint and even Outlook. Below are some brief instructions and examples on some of the popular method used in “Selection” object.

Selection.Move method: collapses the specified selection to its start or end position and then moves the collapsed object by the specified number of units.

e.g., Move the cursor from current position to next line in a MS Word document:

```
Selection.MoveDown unit:=wdLine, Count:=1
```

Selection.StartOf/EndOf method: moves or extends the starting/ending character position of a range or selection to the end of the nearest specified text unit.

e.g., Move the cursor from current position to the start of the current line, and then extend to the end of current paragraph to select all text in this paragraph:

```
Selection.StartOf unit:=wdLine
Selection.EndOf unit:=wdParagraph, Extend:=wdExtend
```

Selection.Find property: returns a Find object that contains the criteria for a find operation.

e.g., Find superscript 2 in a document and replace this text with RTF code:

```
Selection.Find.ClearFormatting
Selection.Find.Font.Superscript = True
Selection.Find.Replacement.Font.Superscript = False
With Selection.Find
  .Text = 2
  .Replacement.Text = "{sup " & Chr(39) & 2 & Chr(39) & "}"
  .Forward = True
  .MatchCase = True
End With
Selection.Find.Execute Replace:=wdReplaceAll
```

Selection.Information property: returns information about the specified selection.

e.g., Check whether the selected text is within a table:

```
If Selection.Information(wdWithInTable) = False Then <...>
```

e.g., Return the current page number of the selected text:

```
endPage = Selection.Information(wdActiveEndPageNumber)
```

Meanwhile, a user-defined add-in is an object that can be attached in MS Word. Once add-in object is attached in MS Word, it will be displayed in the Add-ins sections as a button, which can be directly used every time you open MS Word. Programmer can develop VBA code in the add-ins object to perform some text format check and update in the TLG shell.

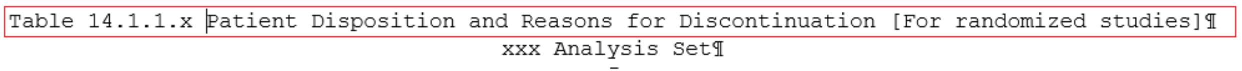
DEFINE STANDARD FORMAT/LAYOUT FOR TLG SHELL

Standardization is always the highest priority for automation. To precisely extract different component in the TLG shell, certain format needs to be specified at first. Table 1 shows a typical example of standard format specification for title number, title name, population set, footnote, programming note.

	Title Number & Name	**Population Set**	**Source & Footnote**	**Programming Note**
Style	Heading 2	Plain text	Plain text	Plain text
Bold	N	N	N	Y
Align	Mid	Mid	Left	Left
End Mark	Carriage Return	NA	Carriage Return	NA

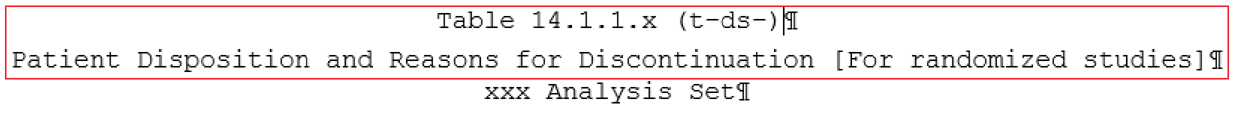
Table 1. Format specification for metadata component in mockup.

Layout is also important for metadata extraction. For example, layout for title/population should follow certain company wise TLG template. Figure 1 and figure 2 show two different layout for the title/population in TLG shell, and it will affect the auto-extraction if layout change. Figure 3 shows a typical layout of a standard table in TLG shell.



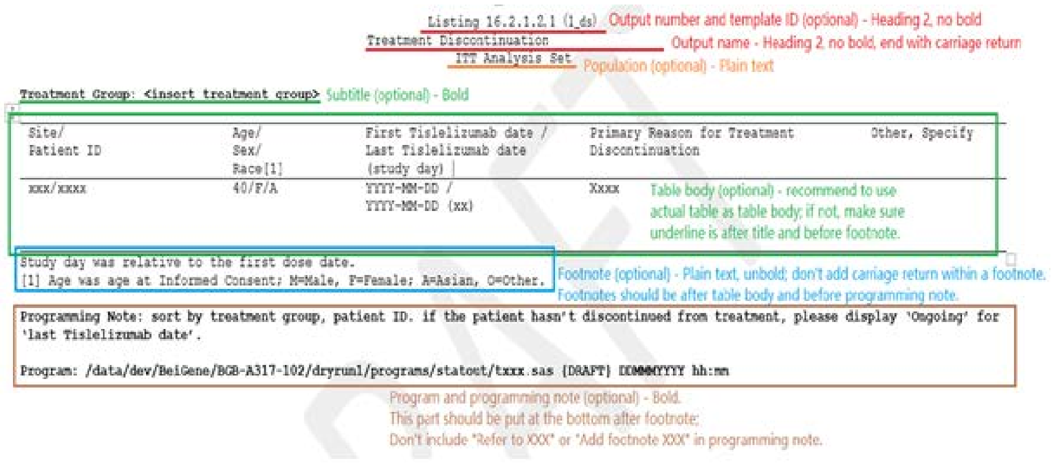
```
Table 14.1.1.x Patient Disposition and Reasons for Discontinuation [For randomized studies]¶
                                    xxx Analysis Set¶
```

Figure 1. Title number and name are placed in the same paragraph



```
                          Table 14.1.1.x (t-ds-)¶
Patient Disposition and Reasons for Discontinuation [For randomized studies]¶
                             xxx Analysis Set¶
```

Figure 2. Title number and name (along with program ID) are placed in separate paragraphs



```
                        Listing 16.2.1.2.1 (1_ds)
                    Treatment Discontinuation
                        ITT Analysis Set

Treatment Group: <insert treatment group>

Site/          Age/       First Tislelizumab date /   Primary Reason for Treatment   Other, Specify
Patient ID     Sex/       Last Tislelizumab date      Discontinuation
               Race[1]    (study day)
xxx/xxxx       40/F/A     YYYY-MM-DD /                Xxxx
                          YYYY-MM-DD (xx)

Study day was relative to the first dose date.
[1] Age was age at Informed Consent; M=Male, F=Female; A=Asian, O=Other.

Programming Note: sort by treatment group, patient ID. if the patient hasn't discontinued from treatment, please display 'Ongoing' for
'last Tislelizumab date'.

Program: /data/dev/BeiGene/BGB-A317-102/dryrun1/programs/statout/txxx.sas {DRAFT} DDMMMYYYY hh:mm
```

Output number and template ID (optional) - Heading 2, no bold

Output name - Heading 2, no bold, end with carriage return

Population (optional) - Plain text

Subtitle (optional) - Bold

Table body (optional) - recommend to use actual table as table body; if not, make sure underline is after title and before footnote.

Footnote (optional) - Plain text, unbold; don't add carriage return within a footnote. Footnotes should be after table body and before programming note.

Program and programming note (optional) - Bold. This part should be put at the bottom after footnote; Don't include "Refer to XXX" or "Add footnote XXX" in programming note.

Figure 3. Specific format and style for all component of a table shell.

VBA MACRO: IDENTIFY COMPONENTS PER STANDARD FORMAT/LAYOUT

Once the layout and format are specified in company standard level, we need to make sure the specification is fully followed by every study. However, manually check and edit each table, listing, figure in TLG shell is very time consuming and frustrated for either programmer or statistician. Enhance, we need to explore more efficient way of auto-check and update format in study level TLG mock.

Below VBA code snippet shows a way to identify title number/name and highlight the text in Blue:

```
With Selection.Find
  'Identify the start of a table in TFL shell
  .Text = "Table"
  .Font.Bold = False
  .MatchWildcards = True
  .Execute Forward:=True
  Do While .Found
    If Selection.Information(wdActiveEndPageNumber) > tocEndPageNum Then
      'Process with output number
      Selection.EndOf unit:=wdLine, Extend:=wdExtend
      title1 = Selection.Text
      With Selection
        .Text = Trim(Replace(title1, Chr(11), Chr(13)))
        .Style = "Heading 2"
        .Font.Bold = False
        'Highlighted in blue
        .Range.HighlightColorIndex = wdTurquoise
        .ParagraphFormat.Alignment = wdAlignParagraphCenter
      End With

      'Process with output name
      Selection.MoveDown unit:=wdLine, Count:=1
      Selection.StartOf unit:=wdLine
      Selection.EndOf unit:=wdParagraph, Extend:=wdExtend
      title2 = Selection.Text
      With Selection
        .Text = Trim(Replace(title2, Chr(11), Chr(13)))
        .Style = "Heading 2"
        .Font.Bold = False
        'Highlighted in blue
        .Range.HighlightColorIndex = wdTurquoise
        .ParagraphFormat.Alignment = wdAlignParagraphCenter
      End With
    End If
  Loop
End With
```

Similarly, we can move down to identify population set and highlight the text in green:

```
With Selection.Find
  'Identify the start of TFL output
  .Text = "Table[1-3] "
  .Font.Bold = False
  .MatchWildcards = True
  .Execute Forward:=True
  Do While .Found
    If Selection.Information(wdActiveEndPageNumber) > tocEndPageNum Then
      'Process with output number
      <...>
```

```
        'Process with output name
        <...>
        'Process with population
        Selection.MoveDown unit:=wdLine, Count:=1
        Selection.StartOf unit:=wdLine
        Selection.EndOf unit:=wdLine, Extend:=wdExtend
        population = Selection.Text
        If Len(Trim(Selection.Text)) > 2 And Selection.Font.Bold = False Then
          With Selection
            .Style = "Plain Text"
            .Font.Bold = False
            'Highlighted in Green
            .Range.HighlightColorIndex = wdBrightGreen
            .ParagraphFormat.Alignment = wdAlignParagraphCenter
          End With
        End If
      End If
    Loop
  End With
```

For footnote, we can move over to the bottom of the table body and highlight footnote text in yellow. When footnote is scanned, we can move to the identification of the start of the next TFL output (do … loop ...):

```
  With Selection.Find
    'Identify the start of TFL output
    .Text = "Table[1-3] "
    .Font.Bold = False
    .MatchWildcards = True
    .Execute Forward:=True
    Do While .Found
      If Selection.Information(wdActiveEndPageNumber) > tocEndPageNum Then
        'Process with output number/name
        <...>
        'Process with population
        <...>
        'Process with footnote
        Selection.MoveDown unit:=wdLine, Count:=1
        Selection.StartOf unit:=wdLine
        Selection.EndOf unit:=wdParagraph, Extend:=wdExtend
        'Presence of programming note marked as table end
        Do While <the end of this output or the start of next output>
          If Selection.Information(wdWithInTable) = False Then
            a = Len(Trim(Selection.Text))
            If <the end of this output or the start of next output> Then
              Selection.StartOf unit:=wdLine
              Exit Do
            ElseIf a > 2 And Selection.Font.Bold = False Then
              Footnote = Selection.Text
              If InStr(UCase(Replace(Footnote, " ", "")), "SOURCE:") = 0 Then
                With Selection
                  .Text = Trim(Replace(Footnote, Chr(11), ""))
                  .Style = "Plain Text"
                  .Font.Bold = False
                  'Highlighted in Yellow
                  .Range.HighlightColorIndex = wdYellow
                  .ParagraphFormat.Alignment = wdAlignParagraphLeft
                End With
      End If
     End If
    Else
     **<End loop when document moved to end>**
    End If
   Loop
  End If
  'Find next table
  Selection.EndOf unit:=wdParagraph
  With Selection.Find
   .Text = "Table[1-3] "
   .Font.Bold = False
   .MatchWildcards = True
   .Execute Forward:=True
   .ParagraphFormat.Alignment = wdAlignParagraphCenter
  End With
 Loop
End With
```

If the TLG shell exactly follow standard-defined format/layout, the title/population/footnote will be properly identified in the shell (shown as Figure 4). Please note that the “Data Source”, “Program address” and “Programming Note” will not be identified as footnote if the format or wording clearly. (e.g., Programming note use **bold** style font; Data Source start with “Data Source:”)

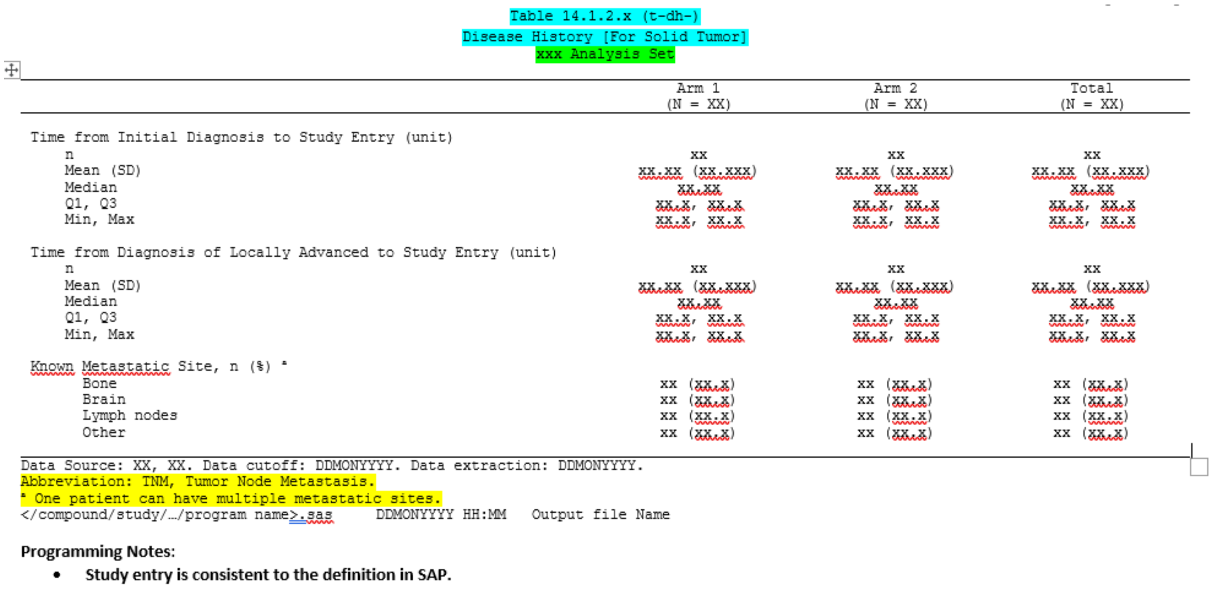


Table 14.1.2.x (t-dh-)
Disease History [For Solid Tumor]
xxx Analysis Set

	Arm 1 (N = XX)	Arm 2 (N = XX)	Total (N = XX)
Time from Initial Diagnosis to Study Entry (unit)			
n	xx	xx	xx
Mean (SD)	xx.xx (xx.xxx)	xx.xx (xx.xxx)	xx.xx (xx.xxx)
Median	xx.xx	xx.xx	xx.xx
Q1, Q3	xx.x, xx.x	xx.x, xx.x	xx.x, xx.x
Min, Max	xx.x, xx.x	xx.x, xx.x	xx.x, xx.x
Time from Diagnosis of Locally Advanced to Study Entry (unit)			
n	xx	xx	xx
Mean (SD)	xx.xx (xx.xxx)	xx.xx (xx.xxx)	xx.xx (xx.xxx)
Median	xx.xx	xx.xx	xx.xx
Q1, Q3	xx.x, xx.x	xx.x, xx.x	xx.x, xx.x
Min, Max	xx.x, xx.x	xx.x, xx.x	xx.x, xx.x
Known Metastatic Site, n (%) [a]			
Bone	xx (xx.x)	xx (xx.x)	xx (xx.x)
Brain	xx (xx.x)	xx (xx.x)	xx (xx.x)
Lymph nodes	xx (xx.x)	xx (xx.x)	xx (xx.x)
Other	xx (xx.x)	xx (xx.x)	xx (xx.x)

Data Source: XX, XX. Data cutoff: DDMONYYYY. Data extraction: DDMONYYYY.
Abbreviation: TNM, Tumor Node Metastasis.
[a] One patient can have multiple metastatic sites.
</compound/study/…/program name>.sas   DDMONYYYY HH:MM   Output file Name

Programming Notes:

- **Study entry is consistent to the definition in SAP.**

Figure 4. Valid component in TLG shell can be properly highlighted

ADD-IN BUTTON: UPDATE INVALID COMPONENTS PER STANDARD FORMAT/LAYOUT

However, if invalid components are found that not properly highlighted in blue, green or yellow, add-in buttons can be applied in MS Word to update title/population/footnote based on user-defined format, respectively (e.g., Format title/population/footnote button). Once the invalid components are updated by the above buttons, Figure 5 shows examples on add-in buttons attached in MS Word.

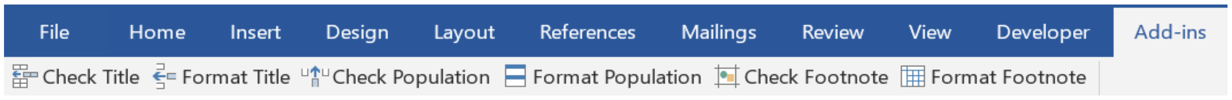


Figure 5. Add-in buttons in MS Word

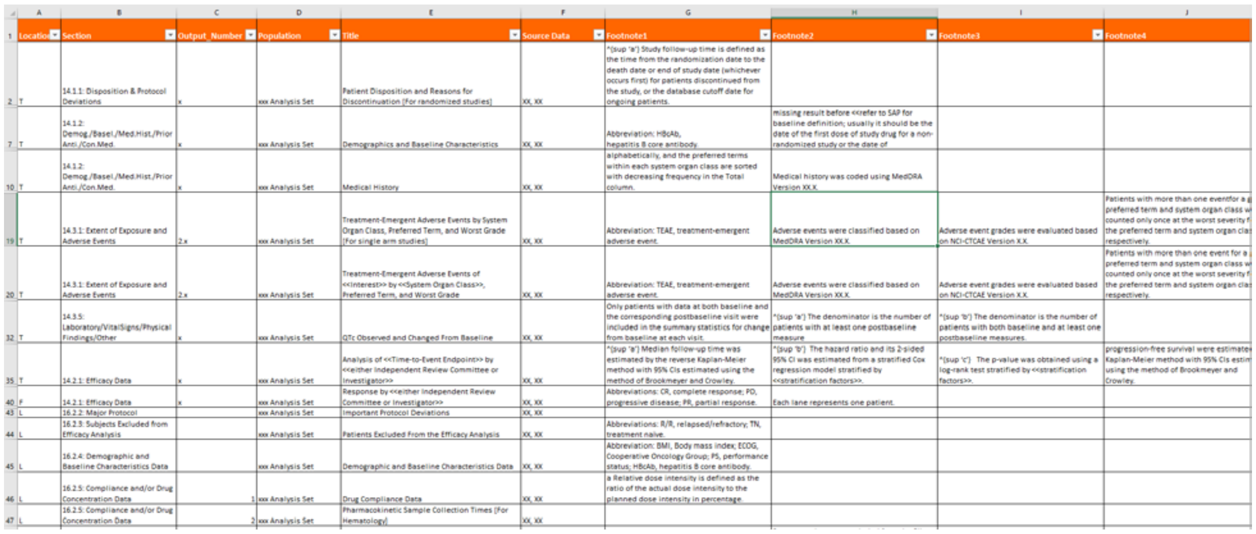
Below code snippet shows an example of user-defined format of metadata component that included in the add-in object:

```
'Define add-in button name:  Check footnote & Format footnote
Dim objButton1, objButton2 As CommandBarButton

Set mybar1 = CommandBars.Add(Name:="Mockup", Position:=msoBarFloating)
mybar1.Visible = True

With mybar1
  Set objButton1 = .Controls.Add(Type:=msoControlButton, Before:=1)
  With objButton1
            .Caption = "Format Footnote"
            .OnAction = "Format_Footnote"
            .Style = msoButtonIconAndCaption
            .FaceId = 300
  End With

  Set objButton2 = .Controls.Add(Type:=msoControlButton, Before:=1)
  With objButton2
    .Caption = "Check Footnote "
    .OnAction = "Check_Footnote"
    .Style = msoButtonIconAndCaption
    .FaceId = 297
  End With
End With

'Function defined: Check footnote
Private Sub Check_Footnote()
  If Selection.Font.Bold = True Or Selection.Style <> "Plain Text" Then
    MsgBox ("Invalid footnote, please use ""Format Footnote"" button.")
  ElseIf Selection.Information(wdWithInTable) Then
    MsgBox ("Footnote is within the table, please update per standard.")
  Else
    MsgBox ("Footnote format is valid!")
  End If
End Sub

'Function defined: Format footnote
Private Sub Format_Footnote()
  Dim footnote As String
  Dim footnote_ As String
  Dim intLenOfString As Integer
  Dim footnote_num As Integer
  footnote = Selection.Text

  intLenOfString = Len(footnote)
  For i = 1 To intLenOfString
    Select Case Mid(footnote, i, 1)
    Case Chr(13)
      footnote_num = footnote_num + 1
    End Select
  Next i

  For j = 1 To footnote_num
    Selection.StartOf Unit:=wdParagraph
    Selection.EndOf Unit:=wdParagraph, Extend:=wdExtend
```

```
        footnote_ = Selection.Text
        If InStr(UCase(Replace(footnote_, " ", "")), "SOURCE:") = 0 Then
          With Selection
            .Text = Trim(Replace(footnote_, Chr(11), ""))
            .Style = "Plain Text"
            .Font.Bold = False
            .Font.Italic = False
            .Font.Underline = False
            .ParagraphFormat.Alignment = wdAlignParagraphLeft
          End With
        End If
        Selection.MoveRight Unit:=wdCharacter, Count:=1
      Next j
    End Sub
```

EXTRACT COMPONENT TO PROGRAM TRACKER

When all the metadata components are validated from TLG shell, the extraction process from shell to tracker can be initiated. Figure 6 shows an example of program tracker that auto generated.



	A	B	C	D	E	F	G	H	I	J
1	Location	Section	Output_Number	Population	Title	Source Data	Footnote1	Footnote2	Footnote3	Footnote4
2	T	14.1.1: Disposition & Protocol Deviations	x	xxx Analysis Set	Patient Disposition and Reasons for Discontinuation [For randomized studies]	XX, XX	^{sup 'a'} Study follow-up time is defined as the time from the randomization date to the death date or end of study date (whichever occurs first) for patients discontinued from the study, or the database cutoff date for ongoing patients.			
7	T	14.1.2: Demog./Basel./Med.Hist./Prior Anti./Con.Med.	x	xxx Analysis Set	Demographics and Baseline Characteristics	XX, XX	Abbreviation: HBcAb, hepatitis B core antibody.	missing result before <<refer to SAP for baseline definition; usually it should be the date of the first dose of study drug for a non-randomized study or the date of		
10	T	14.1.2: Demog./Basel./Med.Hist./Prior Anti./Con.Med.	x	xxx Analysis Set	Medical History	XX, XX	alphabetically, and the preferred terms within each system organ class are sorted with decreasing frequency in the Total column.	Medical history was coded using MedDRA Version XX.X.		
19	T	14.3.1: Extent of Exposure and Adverse Events	2.x	xxx Analysis Set	Treatment-Emergent Adverse Events by System Organ Class, Preferred Term, and Worst Grade [For single arm studies]	XX, XX	Abbreviation: TEAE, treatment-emergent adverse event.	Adverse events were classified based on MedDRA Version XX.X.	Adverse event grades were evaluated based on NCI-CTCAE Version X.X.	Patients with more than one eventfor a preferred term and system organ class w counted only once at the worst severity f the preferred term and system organ clas respectively.
20	T	14.3.1: Extent of Exposure and Adverse Events	2.x	xxx Analysis Set	Treatment-Emergent Adverse Events of <<Interest>> by <<System Organ Class>>, Preferred Term, and Worst Grade	XX, XX	Abbreviation: TEAE, treatment-emergent adverse event.	Adverse events were classified based on MedDRA Version XX.X.	Adverse event grades were evaluated based on NCI-CTCAE Version X.X.	Patients with more than one event for a preferred term and system organ class w counted only once at the worst severity f the preferred term and system organ clas respectively.
32	T	14.3.5: Laboratory/VitalSigns/Physical Findings/Other	x	xxx Analysis Set	QTc Observed and Changed From Baseline	XX, XX	Only patients with data at both baseline and the corresponding postbaseline visit were included in the summary statistics for change from baseline at each visit.	^{sup 'a'} The denominator is the number of patients with at least one postbaseline measure	^{sup 'b'} The denominator is the number of patients with both baseline and at least one postbaseline measures.	
35	T	14.2.1: Efficacy Data	x	xxx Analysis Set	Analysis of <<Time-to-Event Endpoint>> by <<either Independent Review Committee or Investigator>>	XX, XX	^{sup 'a'} Median follow-up time was estimated by the reverse Kaplan-Meier method with 95% CIs estimated using the method of Brookmeyer and Crowley.	^{sup 'b'} The hazard ratio and its 2-sided 95% CI was estimated from a stratified Cox regression model stratified by <<stratification factors>>.	^{sup 'c'} The p-value was obtained using a log-rank test stratified by <<stratification factors>>.	progression-free survival were estimate Kaplan-Meier method with 95% CIs estim using the method of Brookmeyer and Crowley.
40	F	14.2.1: Efficacy Data	x	xxx Analysis Set	Response by <<either Independent Review Committee or Investigator>>	XX, XX	Abbreviations: CR, complete response; PD, progressive disease; PR, partial response.	Each lane represents one patient.		
43	L	16.2.2: Major Protocol		xxx Analysis Set	Important Protocol Deviations	XX, XX				
44	L	16.2.3: Subjects Excluded from Efficacy Analysis		xxx Analysis Set	Patients Excluded From the Efficacy Analysis	XX, XX	Abbreviations: R/R, relapsed/refractory; TN, treatment naive.			
45	L	16.2.4: Demographic and Baseline Characteristics Data		xxx Analysis Set	Demographic and Baseline Characteristics Data	XX, XX	Abbreviation: BMI, Body mass index; ECOG, Cooperative Oncology Group; PS, performance status; HBcAb, hepatitis B core antibody.			
46	L	16.2.5: Compliance and/or Drug Concentration Data	1	xxx Analysis Set	Drug Compliance Data	XX, XX	a Relative dose intensity is defined as the ratio of the actual dose intensity to the planned dose intensity in percentage.			
47	L	16.2.5: Compliance and/or Drug Concentration Data	2	xxx Analysis Set	Pharmacokinetic Sample Collection Times [For Hematology]	XX, XX				

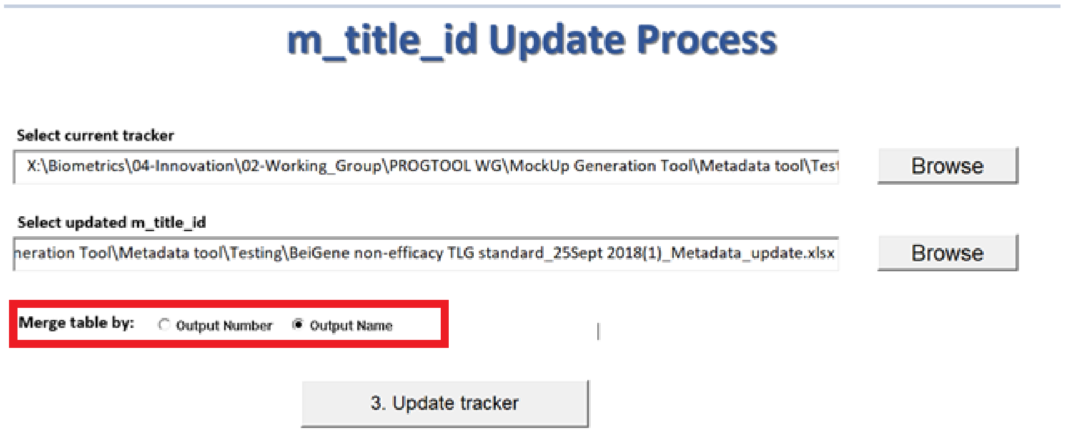
Figure 6. An example of TLG program tracker.

Please note that column B “Section” is based on the title number, so the number should be defined per the CSR section number. Column G-I “Footnote” can be extracted separately if multiple footnotes in TLG shell is separated by carriage return. And if the number of footnotes in a single table/listing/figure exceeds the footnote column limitation in the tracker, e.g., 10 footnotes in shell but only 8 footnote columns in tracker, footnote #8, 9, 10 can be placed together in the last footnote column and separated by “\”.

UPDATE EXISTING PROGRAM TRACKER

Now we get the initial tracker generated per the TLG shell. That is cool! However, how can we quickly update the tracker when **a lot of** updates on the TLG shell in the middle of a CSR, especially when we already filled extra information in the tracker (e.g., QC comments, DEV/QC programmer name, programming status)? It would be painful if we manually copy the extra info from old tracker to the new one, row by row. In that case, a tracker update macro is necessary if we can use certain key information to merge back extra info from old tracker.

First, we need to create a new tracker follow above process. And by using either title name or title number as merge key, we can bring extra info from old tracker to new.


m_title_id Update Process

Select current tracker

X:\Biometrics\04-Innovation\02-Working_Group\PROGTOOL WG\MockUp Generation Tool\Metadata tool\Test

Select updated m_title_id

heration Tool\Metadata tool\Testing\BeiGene non-efficacy TLG standard_25Sept 2018(1)_Metadata_update.xlsx

Merge table by: Output Number [✓ Output Name]

Figure 7. An example of program tracker update process

Below VBA macro is an example on merge existing tracker with the new tracker:

```
Sub Metadata_update_Click()

<Initialization on variables and Excel worksheet>

'Use title name as key to update tracker
If OptionButton1.Value = True Then
  mergeKey = 33
'Use title number as key to update tracker
ElseIf OptionButton2.Value = True Then
  mergeKey = 5
End If

For i = 2 To sht2.UsedRange.Rows.Count
  For j = 2 To sht1.UsedRange.Rows.Count
    If sht1.Cells(j, mergeKey) = sht2.Cells(i, mergeKey) And
sht1.Cells(j, 2) = sht2.Cells(i, 2) Then
      sht2.Cells(i, 6) = sht1.Cells(j, 6) ' source data
      For k = 15 To 23
        'Program Name
        'Output Name Risk Level
        'Programmer QC Method
        'Programmer QC Programmer
        'QC Program Name
        'QC Status
        'Overall Status
        'QC Completion Date
        'Comments
        sht2.Cells(i, k) = sht1.Cells(j, k)
      Next k
      Exit For
    End If
  Next j
Next i

wb_tracker.Close SaveChanges:=False
Set appExcel = Nothing
MsgBox ("Tracker update is done.")
End Sub
```

SPECIAL CASES IN TLG SHELL

SPECIAL CHARACTER HANDLING

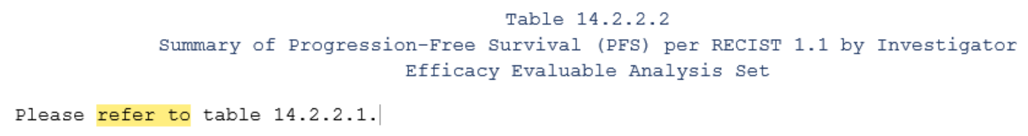
Table 2 shows some examples of special characters (e.g., Superscript for footnote, special operators) that typically presented in the TLG shell. We better replace these characters as Unicode in tracker.

Special character	Unicode	Replacement
Superscript 0-9 A-Z a-z		(*ESC*){sup '[0-9a-zA-Z]'}
Subscript 0-9 A-Z a-z		(*ESC*){sub '[0-9a-zA-Z]'}
Alpha α	U+03B1	(*ESC*){unicode alpha}
Beta β	U+03B2	(*ESC*){unicode beta}
Delta Δ	U+0394	(*ESC*){unicode delta_u}
o with Diaeresis ö	U+00F6	(*ESC*){unicode '00F6'x}
GE ≥	U+2265	(*ESC*){unicode '2265'x}
LE ≤	U+2264	(*ESC*){unicode '2264'x}
NE ≠	U+2260	(*ESC*){unicode '2260'x}

Table 2. Reference table for special character Unicode replacement

FOOTNOTE REFER TO OTHER OUTPUT

Please specify certain layout if one table in TLG shell refer to other TLG. Generally, we need to specify the title number of the output that being referred (Figure 8). In below example, table 14.2.2.2 will share the same footnote with table 14.2.2.1. However, it is better to provide certain WARNING massage or highlighted cell in tracker for user to double check whether the referred footnote is appropriate.



```
                         Table 14.2.2.2
   Summary of Progression-Free Survival (PFS) per RECIST 1.1 by Investigator
                    Efficacy Evaluable Analysis Set

Please refer to table 14.2.2.1.
```

Figure 8. Example of referred table layout in TLG shell

CONCLUSION

This paper introduces the general process on auto extracting metadata component like title, population, footnote, source data, etc. from a standardized TLG shell and then properly placing them into program tracker. To give end user better experience on the process, some user-friendly optimizations on mockup format/layout standardization and tracker update are also introduced. At last, some special cases from TLG shell are discussed and certain solutions are provided.

With actual VBA macro example attached as well, we hope this paper will provide a valuable reference on handling the metadata extraction from TLG shell to program tracker. Moreover, we would be very happy to hear from any talented experts in pharmaceutical industry to explore more efficient methods for this certain process.

REFERENCES

Microsoft Official document "Selection object (Word)" Accessed May 23, 2019. https://docs.microsoft.com/en-us/office/vba/api/word.selection.

Microsoft Official document "AddIn object (Word)" Accessed March 29, 2019. https://docs.microsoft.com/en-us/office/vba/api/excel.addin.

CONTACT INFORMATION

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

Jiapeng He
BeiGene
jiapeng.he@beigene.com
https://www.linkedin.com/in/jiapeng-he-6b347992/

Tingting Zeng
BeiGene
tingting.zeng@beigene.com