DX Review Documentation

Version 1.0.5, August 10, 2023

1. Program Overview

Introduction to DX Review

DX Review is a lightweight Visual Basic module for Microsoft Excel, created to process and beautify ProjNet XML comment reports from DrChecks.

The program processes the XML file(s) gathered by the user, extracting only the relevant data and organizing it into Excel worksheets by DrChecks Review. It summarizes all comments with the evaluations and backchecks in a cleaner and more readable format than Excel does by default. It also provides the user with a region of cells for making comments, noting action items, and assigning responsibilities.

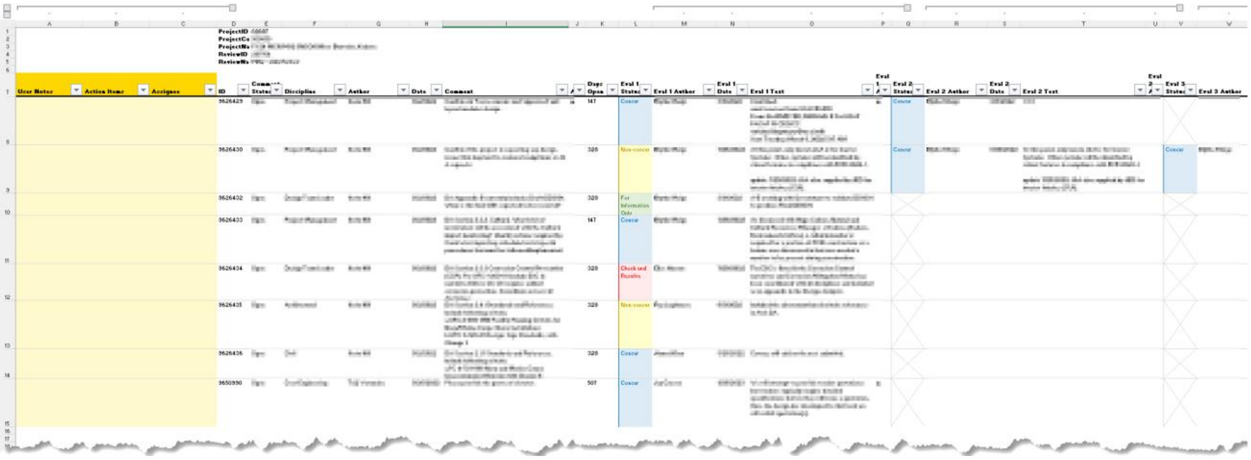


Figure 1. Example Screenshot of Summary

The program will process individual review XMLs, or batch process all review XMLs in a selected folder. The program provides a clean and consistent look; removes the burden of processing and formatting from the user; and helps the user to focus on the real goal: *coordinating and closing comments!*

DX Review is intended to replace the previous iteration of this concept, *DoctorDoctor* (no longer publicly available). *DoctorDoctor* was written in C# as a Windows Forms App; it was determined that it's better to have a simple push-button Excel interface, rather than a standalone application.

Disclaimer

All uses of the terms **DrChecks** and **ProjNet** in this documentation are recognized as trademarks of the respective trademark holder, *Engineer Research and Development Center* (ERDC), who is in no way affiliated with the DX Review project or development team.

For additional information on ProjNet, please visit http://www.projnet.com/index.php on the *National Institute of Building Sciences* website.

Use and Distribution

DX Review was created by Ben Fisher, PE, SE, for *free use and distribution* under the terms of the GNU General Public License v3.0 (license terms posted on the project website: https://github.com/benstanfish/DX-Review/blob/main/LICENSE).

Ben Fisher, PE, SE reserves all rights to the original code, as posted on the project website: https://github.com/benstanfish/DX-Review

Dependencies

This application requires the following libraries, which are often distributed with Microsoft Office products:

- Microsoft XML v6.0 (msxml6.dll)
- Microsoft Scripting Runtime (scrrun.dll)

For users without these libraries installed on their machine, the .dlls can be found in the dependencies folder of the project website: https://github.com/benstanfish/DX-Review/tree/main/dependencies.

These libraries are typically located in the `C:\\Windows\System32\` directory. The user must set a reference to them in the VBA IDE. This process will be described in greater detail in the **Troubleshooting** section.

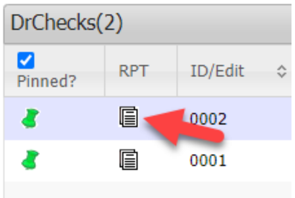
2. Quick Start Guide

Localize Your Review XML Files

The first step in the process begins outside DX Review and Microsoft Excel. You need to log into your ProjNet account and download the XML reports. Here is the process:

1. Select the appropriate project from your assigned projects list.

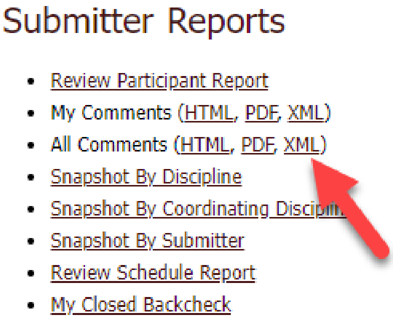
2. In the DrChecks region, click the file icon in the *RPT* column for the Review(s) you want to download.



DrChecks(2)

☑ Pinned?	RPT	ID/Edit
		0002
		0001

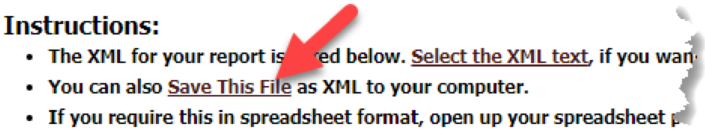
3. Under the Submitter Reports, you'll want to select the XML link in the line *All Comments*. It's recommended that you export All Comments, not just My Comments (although that is possible too) – it's easy enough to filter the comments you don't want to see in the output summary workbook.



Submitter Reports

- Review Participant Report
- My Comments (HTML, PDF, XML)
- All Comments (HTML, PDF, XML)
- Snapshot By Discipline
- Snapshot By Coordinating Discipli
- Snapshot By Submitter
- Review Schedule Report
- My Closed Backcheck

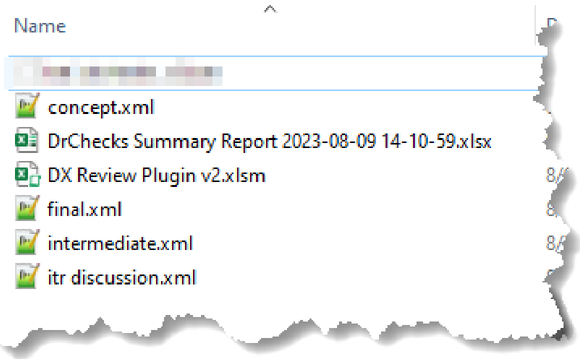
4. In the next screen, simply click the *Save This File* link in the second bullet point.



Instructions:

- The XML for your report is ed below. Select the XML text, if you wan
- You can also Save This File as XML to your computer.
- If you require this in spreadsheet format, open up your spreadsheet p

Each you save a file from the ProjNet website, it will give the XML file a vanilla name. From the standpoint of DX Review, it doesn't matter what the file's name is. The program names the Worksheets based on metadata inside the file. So, name the files in a way that makes sense to you and doesn't overwrite other files (**Figure 2**).



Name

[redacted]

concept.xml

DrChecks Summary Report 2023-08-09 14-10-59.xlsx

DX Review Plugin v2.xlsm

final.xml

intermediate.xml

itr discussion.xml

Figure 2. Example Folder with Multiple XML Files

It's recommended that you batch process multiple phases of the project at the same time – each review is added to the output summary workbook as a different tab. This allows you to review the progress much more easily.

Run the DX Review Macro

The DX Review applet has two macros exposed for the user to use, however the latter (DX_Review_Select_Folder is the preferred/recommended macro):

- DXReview_Select_File
- DXReview_Select_Folder

Technically speaking, the "Select_Folder" method will process single files, if a folder only has a single valid report XML file in them. If the XML file is not formatted using ProjNet's schema, the macro will basically walk away.

There are currently a few ways you can access and run the DX Review macro(s). You can access any of the three from the project website: https://github.com/benstanfish/DX-Review. Currently the recommended method is to download the **DX Review Plugin v2.xlsm** file and use that. As updates are made to the code, that file is maintained (so it's always current).

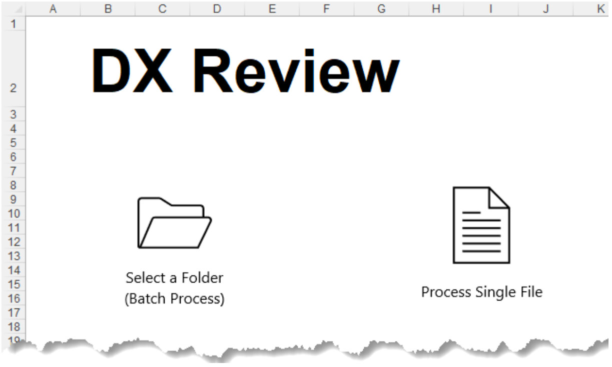


Figure 3. View of the Buttons in the Macro-Enabled Workbook (*DX Review Plugin v2.xlsm*)

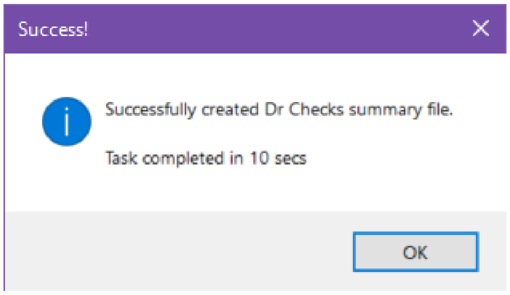
Simply click one of the buttons to run the Macro. Note: The **Select a Folder** version is recommended as its batch processes all valid XML reports at once. When you choose this option, you won't see the files in the dialog box – this is expected.

Coming Enhancement

We're working on creating a better deployment approach, whether that's a ribbon plugin, or a Python app that runs Excel in the background. We've gone away from WinForms apps because some offices do not allow "unknown" apps to be installed. For now, most people can use macro-enabled Excel worksheets. If you don't trust it, review the .bas file on Github or run the .xlsm file in Windows Sandbox.

Output File

Once the program finishes processing the XML files, you'll get a success message (**Figure 4**). Typical run times for multiple files with several hundred comments, evaluations and backchecks each are around 5 – 15 seconds.



Successfully created Dr Checks summary file.

Task completed in 10 secs

Figure 4. Successful Run Message

The program writes a new Excel workbook with the summary output into the same folder as the XML files. It appends a timestamp so that it doesn't overwrite previous runs. The program will automatically navigate to this folder in your explore after the file is saved and closed, but you won't be able to open it or do anything else in Excel until you click OK in the dialog box shown in **Figure 4** (I didn't want to *send keys* to automatically dismiss the message box after some elapsed time – that might be seen by some as malicious codes).

User Note

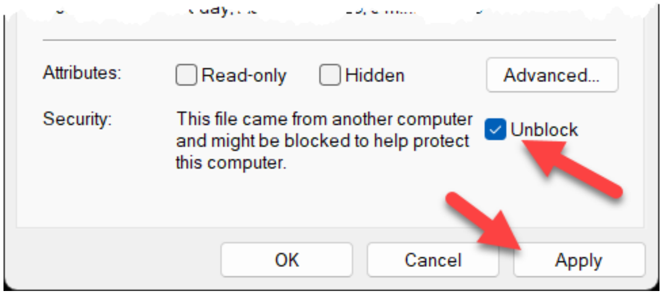
The DX Review VBA code module does not have to be located in each file, nor does it have to be inside the output summary workbook. In fact, the code is specifically designed to create the output summary workbook as a standard Excel workbook (not macro-enabled) so that it can be transmitted around without upsetting people's email filters, etc.

Troubleshooting

Read Only or Blocked Macro-Enabled Workbook

If you've downloaded the macro-enabled Workbook () from the Github site and want to run the code from there, some machines (Windows 11) maybe initially "block" the file – permanently putting it into a read-only state.

To check this, navigate to the file in your explorer, right-click and select properties. Near the bottom of the properties tab, you'll see a "Security..." message and a checkbox that say "Unblock" – make sure that is *checked*. Click apply and it will disappear, after which you will be able to open and run the file like normal.



Attributes: ☐ Read-only ☐ Hidden

Security: This file came from another computer and might be blocked to help protect this computer. ☑ Unblock

Figure 5. Unblock the Downloaded File in Properties

When you open the file, you may also need to give permission to enable the macros.

Missing Dependencies

If you do not see the success message or the output file looks like it's only half-baked, you may be missing some of the **Dependencies**. This code references to two very commonly distributed Microsoft code libraries, which you can find on project webpage: https://github.com/benstanfish/DX-Review/tree/main/dependencies

1. Navigate to the project github link above.

2. Right-click and save the msxml6.dll, msxml6r.dll and scrrun.dll files. It's preferable to save them with all the other major Microsoft libraries in the following folder `C:\\Windows\System32\` (note that if the save as dialog warns you about overwriting, you already have the .dll on your system – you just need to repair the reference… next step)

3. If your administrator does not allow you to save the .dlls to the folder above, you can basically save anywhere and do the next step.

4. Open the **DX Review Plugin v2.xlsm** file. Press `Alt + F11` to open the VBA editor.

5. In the Project Explorer (if not shown press `Ctrl + R`), then select the Excel file In **Tools** menu select the **References…** option (**Figure 6**).

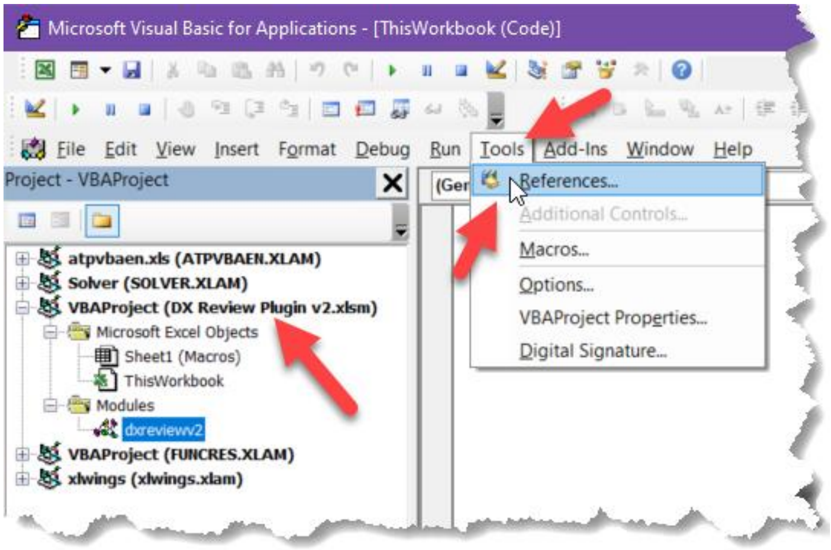
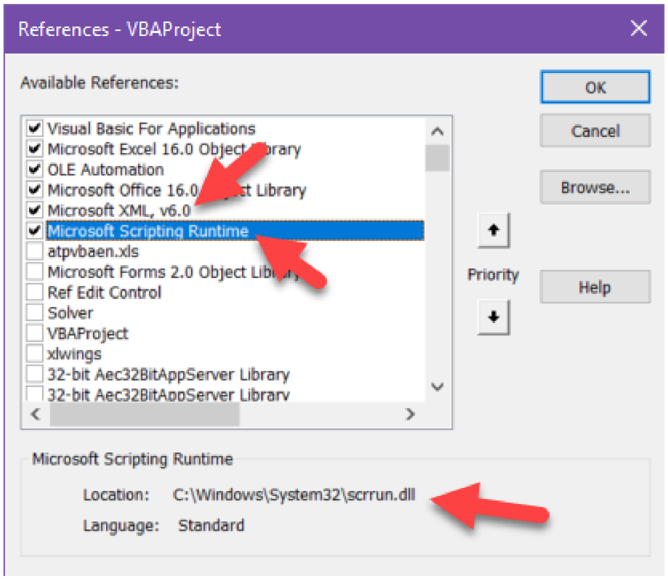


Figure 6. VBA Editor

6. Next in the References dialog, make sure you have the checkboxes next to Microsoft XML, v6.0 and Microsoft Scripting Runtime checked. You may have to scroll through the list to find them. Notice that you can see the location of the .dll file at the bottom of the dialog box.



Available References:

☑ Visual Basic For Applications
☑ Microsoft Excel 16.0 Object Library
☑ OLE Automation
☑ Microsoft Office 16.0 Object Library
☑ Microsoft XML, v6.0
☑ Microsoft Scripting Runtime
□ atpvbaen.xls
□ Microsoft Forms 2.0 Object Library
□ Ref Edit Control
□ Solver
□ VBAProject
□ xlwings
□ 32-bit Aec32BitAppServer Library
□ 32-bit Aec32BitAppServer Library

Microsoft Scripting Runtime

Location: C:\Windows\System32\scrrun.dll
Language: Standard

7. On most systems these libraries are already installed, so you probably only need to set the reference. If the library is not located in `C:\\Windows\System32\` you can click the Browse… button to navigate to the .dll location

8. Register each .dll library by pressing `Windows + R` to open the run command dialog. Type in `CMD` and then hold `Ctrl + Shift + Enter` to run the open command prompt as *administrator*.

9. In the command prompt, type `cd\` and press `enter`

10. Next type in `regsvr32 scrrun.dll` or `regsvr32 msxml6.dll` and press enter

It’s not expected that you’ll have to do the above steps, but if you do – it should only be necessary once.

Snapshot in Time

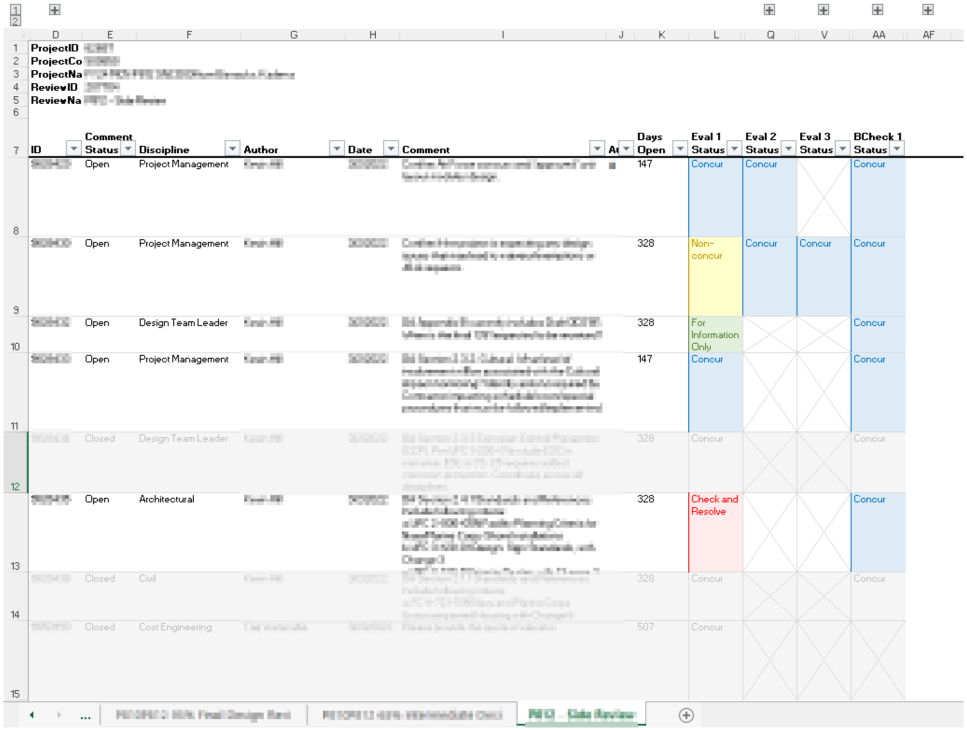
The DrChecks data that is shown in each Worksheet tab is neither linked to the processed XML file(s), nor any data source online. In other words, the output file represents a snapshot in time. It is recommended that you run the app each time you want to make an updated version of the output.

Unfortunately, currently ProjNet does not have a known way to batch upload comments, but this spreadsheet does offer the user a clean way to stage and preserve their responses.

3. Overview of the Output File

Initial View

The output file contains one or more tabs (Worksheets), named based on project information found inside the XML file(s). Each Worksheet has the same layout consisting of several regions. When the file is first opened, it appears similar to **Figure 7**; several of the regions are collapsed.



ProjectID [redacted]
ProjectCo [redacted]
ProjectNa [redacted]
ReviewID [redacted]
ReviewNa [redacted]

ID	Comment Status	Discipline	Author	Date	Comment	At	Days Open	Eval 1 Status	Eval 2 Status	Eval 3 Status	BCheck 1 Status
[redacted]	Open	Project Management	[redacted]	[redacted]	[redacted]		147	Concur	Concur		Concur
[redacted]	Open	Project Management	[redacted]	[redacted]	[redacted]		328	Non-concur	Concur	Concur	Concur
[redacted]	Open	Design Team Leader	[redacted]	[redacted]	[redacted]		328	For Information Only			Concur
[redacted]	Open	Project Management	[redacted]	[redacted]	[redacted]		147	Concur			Concur
[redacted]	Closed	Design Team Leader	[redacted]	[redacted]	[redacted]		328	Concur			Concur
[redacted]	Open	Architectural	[redacted]	[redacted]	[redacted]		328	Check and Resolve			Concur
[redacted]	Closed	Civil	[redacted]	[redacted]	[redacted]		328	Concur			Concur
[redacted]	Closed	Cost Engineering	[redacted]	[redacted]	[redacted]		507	Concur			

Figure 7. Typical View of Summary Worksheet

Near the top of the sheet (**Figure 8**), you'll noticed small "plus" boxes – these are buttons that expand and collapse the grouped regions. You can also click the "2" at the far left to expand and all and the "1" to collapse all.

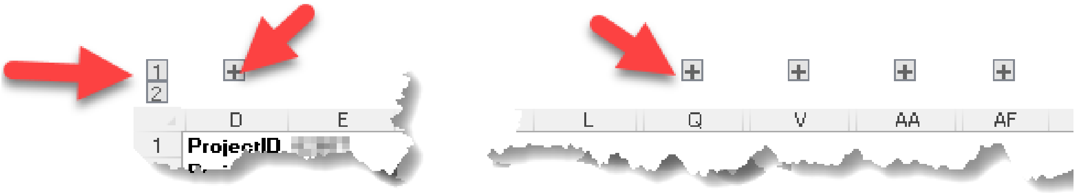


Figure 8. Expand and Collapse Hidden Regions

These "regions" are created using the **Data | Outlining | Group** function in Excel; you can modify as necessary.

Introduction to the Worksheet Report Regions

Once fully expanded (**Figure 9**), I can give you a tour of the various regions in each Worksheet.

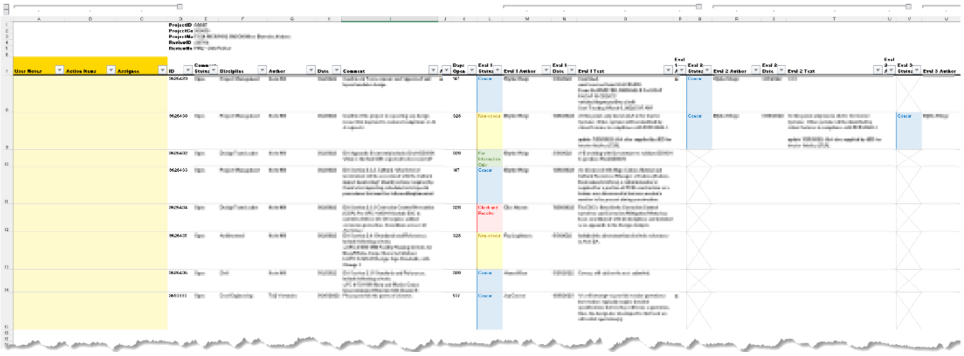


Figure 9. Fully Expanded Example Report

The output consists of four or five regions (depending on how you count). The first region is the smallest, but not the least: the *Project Info Region* (**Figure 10**), which contains the project metadata from the XML file.

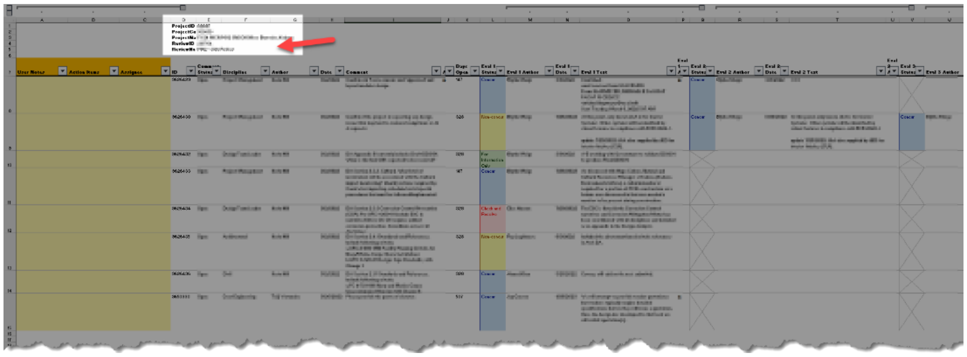


Figure 10. Project Info (Metadata) Region

Initially collapsed, the *User Region* (colored yellow like a Post-It note) is located to the left side of the Worksheet (**Figure 11**). This region is intended to be used for you to keep notes, put draft responses, outline action items, or assign responsibilities.

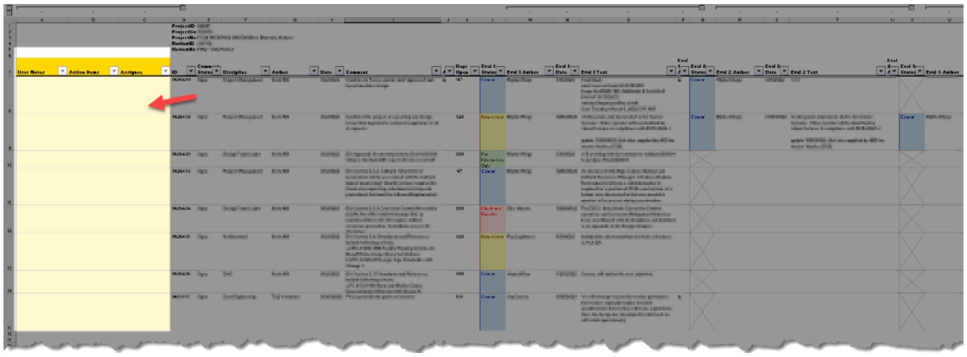


Figure 11. User Region

Left of the User Region, and initially the leftmost visible region is the *Comments Region* (**Figure 12**) with is the pulsating heart of each Worksheet. All comments are listed by the DrChecks ID number. Conditional formatting is applied based on the comment `Status`, whether *Open* or *Closed*.

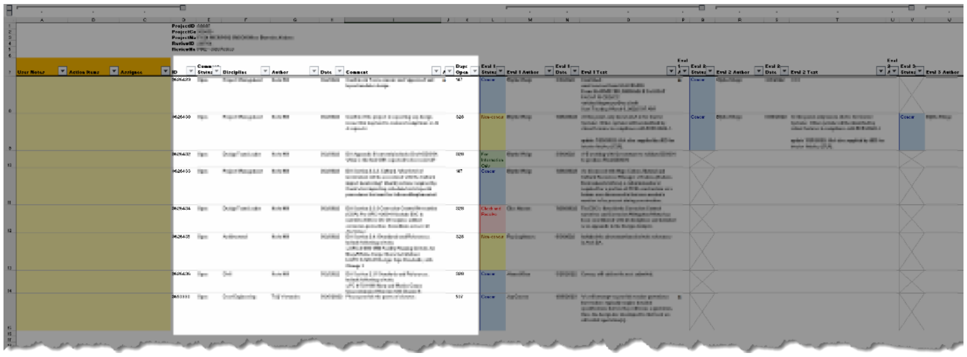


Figure 12. Comment Region

Finally, located to the right had side is the *Response Region* (**Figure 13**), which can be subdivided into the *Evaluations*- and *Backchecks Regions*.

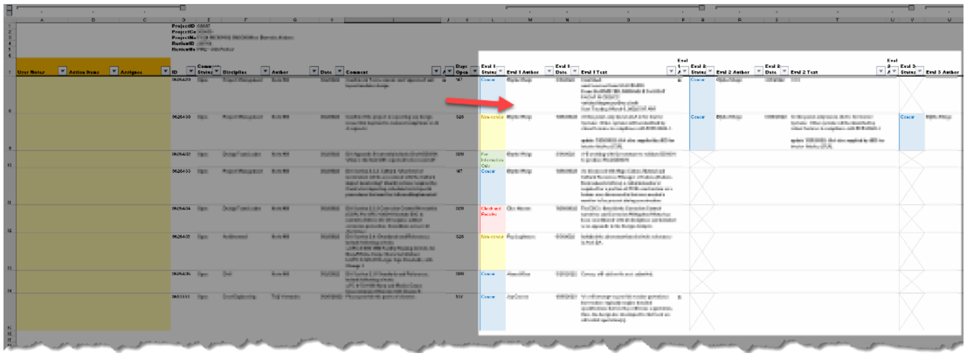


Figure 13. Response (Evaluation and Backcheck) Region

The data populated in the Evaluations and Backchecks regions is always the same, however, the layout of the region depends on how many evaluations and backchecks there are in the XML file. The reasoning behind how this data is structured is discussed in greater detail in the **Structure of Region** section.

Filtering Comment

You'll notice the filter buttons along the header row. These allow you to sort and filter, by data, color, etc.

Conditional Formatting

There are only five conditional formatting rules in each worksheet. The primary condition, mentioned above, relates to whether the comment is opened or closed. If closed, the comment row is muted.

The other three formats relate to the `status` in each evaluation and backcheck column. You can feel free to customize these as you like.

4. Development Notes

Regions of Data within Each Worksheet

Each XML report is summarized into an individual Excel Worksheet (i.e., a “tab” inside the combined Workbook). The data in each Summary sheet can be divided into several regions:

- Project Info (Metadata)
- User Region
- Comments
- Reponses (Evaluations and Backchecks)

Each of these regions is discussed in detail below.

Project Info (Metadata) Region

The root node of the XML summary, called `<ProjNet>`, usually has two child nodes: the first is `<DrChecks>` and the second is `<Comments>`. The `<DrChecks>` node contains the following project metadata properties, which are expanded into a header in the Summary sheet.

- Project ID
- Project Control Number
- Project Name
- Review ID
- Review Name

It is the final element node `<ReviewName>` is what is expanded into the name of the Summary sheet. The name of the review is used as the tab name, which does result in lengthy tab lengths. Excel does not permit tab names longer than 31 characters.

User Region

The following columns are provided in the User Region. This region is located leftmost in a Summary tab to topographically indicate that its data is NOT part of the original XML report content. The following is a list of current fields provided in the User Region

- User Notes
- Action Items
- Assignee

The User Region location and dimensions must not be modified to allow for future enhancements to the plugin.

> **Coming Enhancement**
>
> A future enhancement will allow the user to compare values in the User Region between successive versions of the Summary sheet. The user will need to select Summaries to compare. Need to work out what sort of presentation of version control will be implemented.

Comments Region

As discussed in the **Project Info (Metadata) Region** section, the second child of the `<ProjNet>` root element node is the `<Comments>` element, which contains the

As discussed above, the `<Comments>` node of the XML report contains all the comments, as well as de the *Responses* (a term I am using to include both *evaluations* and *backchecks*). Responses are structured as descendants of a given `<comment>` ancestor node.

> **Developer Note**
>
> When ProjNet exports a report to XML, unfortunately it serializes its data in such a way that the `<comment>` element is both a descendant and ancestor of itself, which is both unnecessary and bad form. Because of this, to get all the comments correctly the XPATH expression of `Comments/comment` must be used.

The Comments region is located immediately to the right of the User Region. The comments region does not include all fields provided in the XML. The following properties are included:

- ID
- Status
- Discipline
- Created By
- Created On
- Comment Text
- Attachment
- ***Days Open***

These properties are not in the same order that they appear in the XML `<comment>` element. There are a few things for the user to know about the data found in several of these fields:

1. The `Status` is field is either *open* or *closed* and is different from the evaluation field by the same name

2. The foremost condition used to format each row of the Worksheet is determined by the `Status` field

3. Rather than list the name of the document provided in the Attachment field, a symbol is shown to indicate that an attachment was associated with the comment. Because the attachment is not exported with the XML report from ProjNet, there's no reason to do more

4. **Days Open** is not a field native to the XML report. It is a calculated field, determined based on the day count from the `Created On` date

> `Days Open` **Property in Comments Region**
>
> The logic of the Days Open field is as follows: if the comment `Status` is *open*, the days elapsed between the time when the the code in this program is run and the `Created On` date is calculated and returned. If the `Status` is *closed*, the program looks at the `Created On` date of the last backcheck child of the `<comment>` and calculates and returns that date difference.

Response Region

If *evaluations* or *backchecks* are present in the XML report, a Response Region is created to the right of the Comments region.

Background

In the context of this program, “Responses” is used to define both *evaluations* and *backchecks*. Each `<comment>` node has `<evaluations>` and `<backchecks>` child elements, which have individual *evaluation* and *backcheck* elements.

> **Developer Note**
>
> Another issue with ProjNet’s XML serialization approach has to do with the `nodeName` of individual *evaluation* or *backcheck* element nodes. In an idea XML schema, each evaluation node would be called `<evaluation>` however, ProjNet appends 1, 2, 3, etc. to each `nodeName`, e.g., `<evaluation1>`, `<evaluation2>`, `<evaluation3>`. This is both unnecessary and makes naïve XPATH queries cumbersome if they are even possible. Backcheck nodes follow the same unfortunate pattern.
>
> To get around this issue, because each *evaluation* or *backcheck* element are children of the `<evaluations>` or `<backchecks>` elements, this program uses XPATH to create an `IXMLDOMSelection` of `<evaluations>` and `<backchecks>` elements, which means the child nodes are all *indexed items*.
>
> How this issue affects naïve traversal of the XML DOM was the major impetus for this major refactor (Version 2), *DoctorDoctor* being the first iteration.

Structure of Region

There are two possible ways to order the Response comments: *structurally*, based on the order they appear in the XML tree, or *chronologically*, based on the `Created On` elements for each comment. In reality, the individually comment children of the `<evaluations>` and `<backchecks>` are ordered chronologically. But there are two disadvantages to ordering all Responses based on chronologically.

First, it does require additional, complex code to combine all the comments into a single pool, then order them chronologically. Second, when summarizing the data in the Reponses region of the Excel worksheet, it means that there is no predictability for which column evaluations or backchecks appear in; this could hamper future development.

For these regions, the Response region, which is subdivided into the Evaluations and Backchecks regions, is ordered first structurally (evaluations, then backchecks) then chronologically (for individually comments within the respective evaluation or backcheck regions).

It’s possible to have any number of evaluations and backchecks. When the program initially parses the XML file, it determines the maximum number of evaluations and backcheck comments for all comments in the file. It then structures the Evaluations and Backchecks regions accordingly.

Presented Data

Similar to the parent `<comment>` not all the fields present in the XML report are provided in the summary. Here is a list of the data included:

- Status
- Created By

- Created On
- Evaluation Text (or Backcheck Text)
- Attachment

Each backcheck element in the XML does reference the related “sibling” evaluation comment (although these are not technically sibling elements because of how ProjNet serializes the XML file). It was debated whether to include the evaluation IDs, how decided that the context will be clear enough and ultimately can be verified in ProjNet.

The `status` field has different values from the status field for a `<comment>`. There are four values:

- Concur
- For Information Only
- Non-concur
- Check and Resolve

The status field of the evaluation comments are used in Excel conditional formatting rules, applied to each comment row then the overall comment status value is *open*.

Coming Enhancement

The color coding for open comments is based on the evaluation `status` parameter. It is possible for there to be more than one open evaluation comment. The current plan is to determine the conditional formatting color based on the highest weighted evaluation status. The proposed weight scale will be *concur* (1) < *for information only* (2) < *non-concur* (3) < *check and resolve* (4).

This weight order is based on stylistics: *non-concur*, without much explanation, indicates disagreement, but that can sometimes be resolved by explanation and no further discussion. However, we have used *check and resolve* to specifically mark comments that the AE team wants to discuss during OBR meetings. In that sense, between these two labels, the latter does seem to connote a degree of active coordination that is required to resolve a disagreement.

The user can override and customize the conditional formation, however, to meet their needs.