GFIT Framework Reporting Structure : How to use ALM reporting effectively with Microsoft Word templates

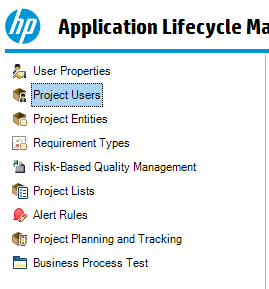
# Introduction

Getting information out of HP ALM is a much misunderstood and maligned subject. I myself have often complained that it gives you “nothing or far, far too much”

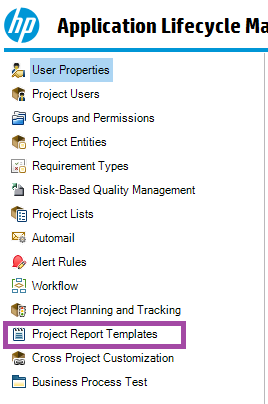
I have had the opportunity recently to take a deep dive into this functionality and get to understand how it hangs together. Like most things ALM, its acutually really powerful when you understand it.

# Getting set up

Lets start with a basic ALM project. You probably do not have report templates in place.



If your project customization screen looks like this example to the left then you don’t have a linked “Template” project. This is where we want to be to be standardising reporting across a number of ALM projects.



So, we create an ALM template project and then we create a satellite project cloned from and linked to that template. That will give us a view that looks a bit like this in the *satellite* project.

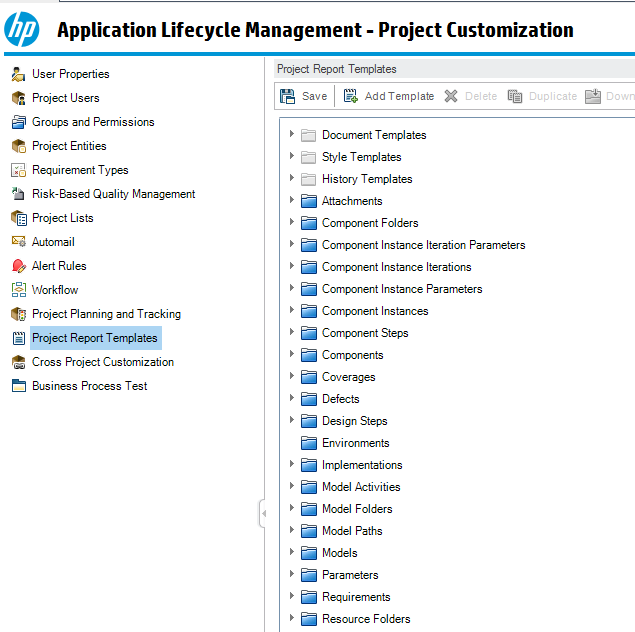
We can see here that the “Project Report Templates“ are available to us. This will most likely contain a large number of default options that we are *not* going to use until we understand them.

So, lets head back over to the *Template* project and take a look at the project reports templates there.

## In the template project, building templates.

The word “template” is much overused, and we need to be careful in what context we understand it at any given instance. Just be aware of this.

So, we are back in the template project in the customization screens. We see something like this:



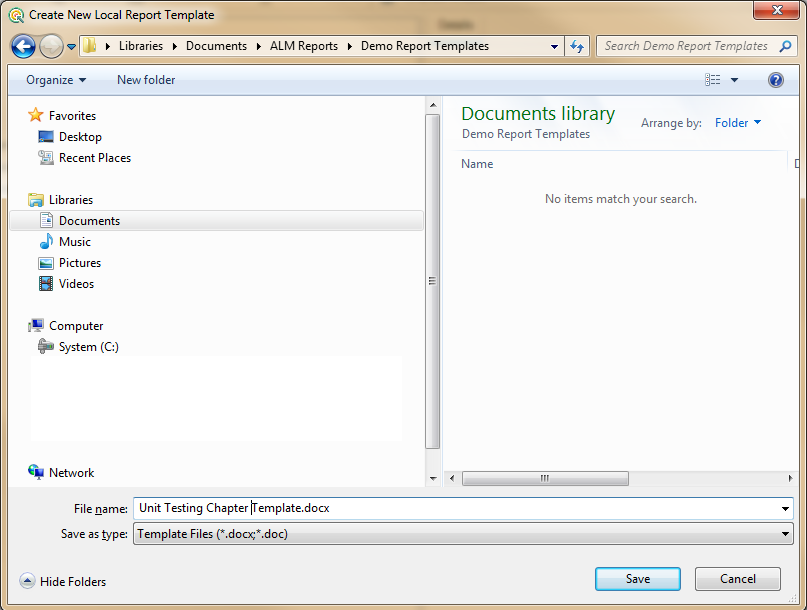
The best way to think of these is as “chapters” in your test report. If you want a standard, cross project test report then you want to have standard chapters. We are going to put together a simple Test Report that covers Unit Test, System Test and UAT as well as defects found & outstanding issues. Very basic.

# The unit test chapter

To create a chapter that talks about unit testing with some boilerplate text we select the template creator button

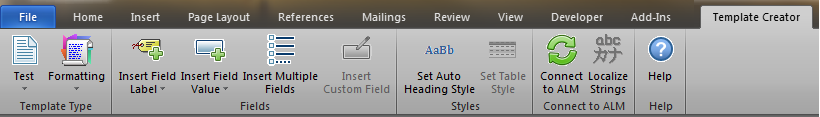


This gets us the following “save” dialog and as you can see I have saved the file as “unit testing chapter template.docx”

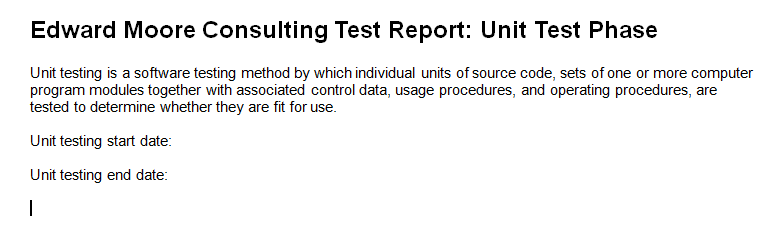


The document will open and depending on local security settings will present a macro enable option. Choose “Enable content” and we can now create our unit testing chapter.

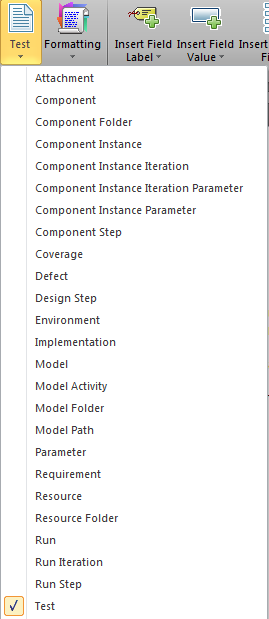
Note that word has a new ribbon option called “Template Creator”



Lets not worry too much about that initially, and get the boilerplate text in place along with any fields that we want the test managers to complete manually. Here I have a start and end date for the unit testing phase. The objective is to limit the amount of data that is not generated automatically from the tool.



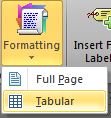
Here I have added a title, a brief description of unit testing from wikipedia to give the report some “bulk” and the two fields that I have left for completion manually. Lets now add the good bits.



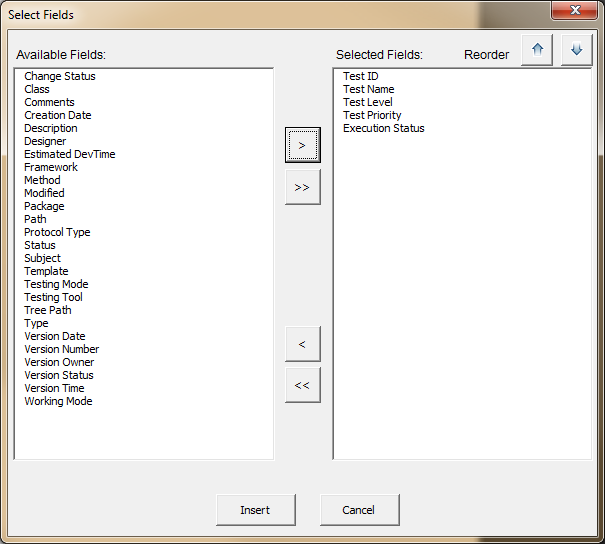
The first thing we need to configure is “what type of entity will this “chapter” cover.” In this case its test so on the template type selection at the left of the ribbon we select “test” from the drop down list of possible entities. *You cannot have a chapter that draws information from two entity types!*

This is why it is important to focus on these reports as being chapters within a larger whole, you are not yet building the whole report, take it slowly and build each chapter carefully.

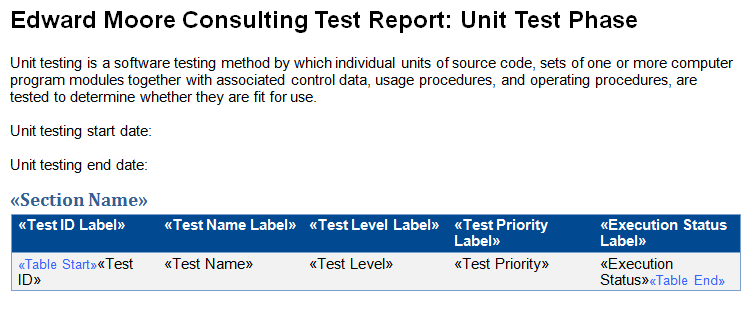
We now want to add the details of this entity, in this case the unit testing. We select the Formatting ribbon option and lets go for a tabular layout



That presents this following dialog box and for my table I want Test ID, Test Name, Test Level ( Although its Unit testing, clearly) Test Priority and execution Status.



Selecting these options and then hitting “insert” will give me a word template that looks a little like this:



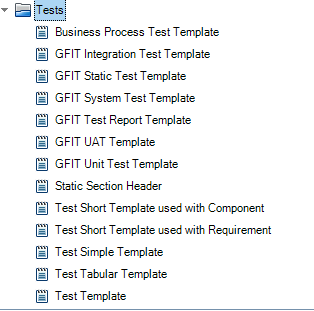
Save and close this file now, don’t attempt to add any more detail from the ALM configuration, you can add more boilerplate if you like.

Repeat this process for system testing and UAT changing the boilerplate text and any other items that you want to be distinct on each of the chapters. This will leave you with three word files.

Open ALM again and head to the shared project templates customization area. We are now going to add the templates we have created to the tree. You cannot create new folders for some reason so we will add them to the “tests” section with the “add template” button



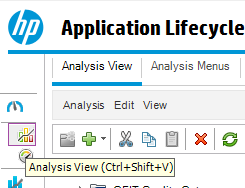
Here’s one I made earlier with the GFIT test templates for each of the test phases.



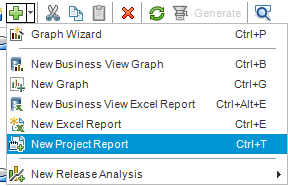
We now need a front page for our report. Follow the same process as used for the test sections to create a front page with your logo, branding and any other text that you want present and standard. Upload this to the document templates section.

We are now done with the components of the report and we can proceed to build it in the reporting dashboard. Head to “Analysis View”

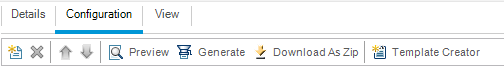
Each project that wants to use the report will have to build it in their project. *The report itself is not exported and shared, only the components*



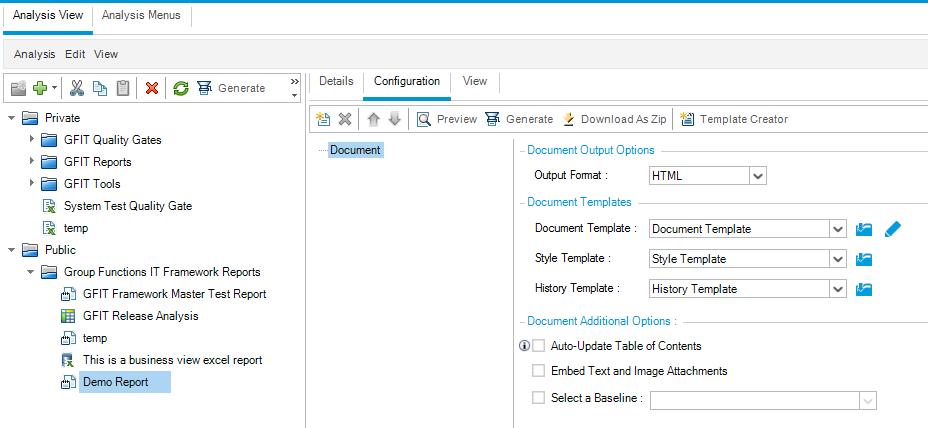
Create a new project report



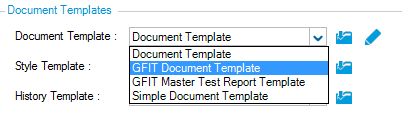
Select configuration



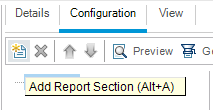
At this point you will have a new, empty report and options to select output format and the document template.



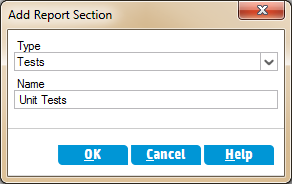
Set the output format to Word and select the document template you created earlier

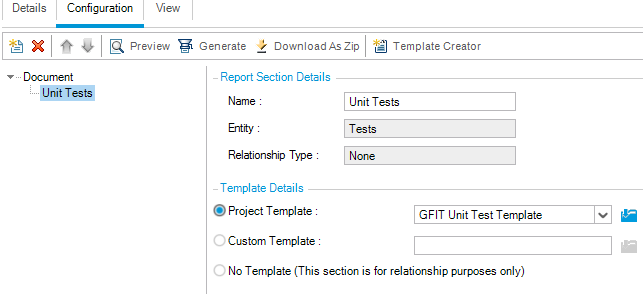


Select “add report section”



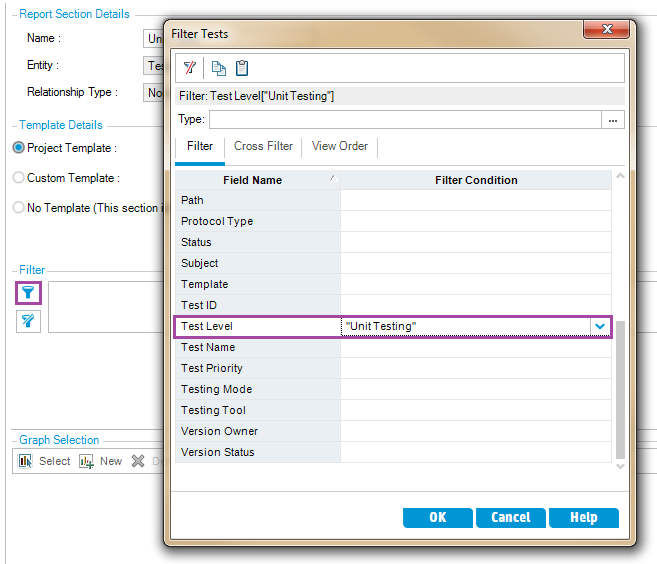
From the dialog select “tests” and give the section a name, initially lets go with “Unit Tests”





Select the unit test template that you created as the template for this section.

Configure the filter to only show unit tests in this section. You can also cross filter and configure view order



Repeat these steps for the System and UAT sections.

The finished report can now be generated by pressing the generate button

